TUGAS AKHIR – TE141599

SISTEM NAVIGASI PADA BALON UDARA MENGGUNAKAN GPS DAN KONTROL LOGIKA *FUZZY*

Dimas Arief Rahman Kurniawan
NRP 2212100171

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Rudy Dikairono, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT – TE141599

AIRSHIP NAVIGATION SYSTEM USING GPS AND FUZZY LOGIC CONTROL

Dimas Arief Rahman Kurniawan
NRP 2212100171

Advisor
Dr. Muhammad Rivai, ST., MT.
Rudy Dikairono, ST., MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

SISTEMNAVIGASI PADA BALON UDARA MENGGUNAKAN GPS DAN KONTROL LOGIKA *FUZZY*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Muhammad Rivai, ST.,MT.	Rudy Dikairono, ST., MT.
NIP. 19690426199031003	NIP. 198103252005011002

SURABAYA
JUNI, 2016

SISTEM NAVIGASI PADA BALON UDARA MENGGUNAKAN GPS DAN KONTROL LOGIKA *FUZZY*

Nama : Dimas Arief Rahman Kurniawan
Pembimbing I : Dr. Muhammad Rivai, ST., MT.
Pembimbing II : Rudy Dikairono, ST., MT.

ABSTRAK

Pengembangan sistem navigasi *Unmanned Aerial Vehicle* (UAV) untuk keperluan pemantauan dan eksplorasi semakin meningkat. Jenis UAV yang sering digunakan untuk keperluan ini adalah *quadcopter* dan pesawat *fixed wing*. *Quadcopter* memiliki bentuk yang ringkas sehingga dapat terbang di tempat yang sempit dan kecil. Sedangkan pesawat *fixed wing* sering digunakan untuk pemetaan area yang luas. Namun penggunaan kedua wahana tersebut kurang cocok untuk keperluan pemantauan dan eksplorasi. Q*uadcopter* memiliki jangka terbang dan jarak tempuh yang pendek. Pesawat *fixed wing* dapat terbang dengan jarak tempuh yang panjang, namun mempunyai kecepatan terbang yang tinggi. Salah satu pilihan UAV alternatif untuk keperluan pemantauan dan eksplorasi adalah balon udara. Balon udara sangat cocok digunakan untuk aplikasi ekplorasi dalam kecepatan dan ketinggian yang rendah. Gaya angkat balon udara menggunakan gas helium, sehingga balon udara dapat terbang lebih lama dan dengan jarak yang jauh. Oleh karena itu, pada tugas akhir ini merancang dan merealisasikan sistem navigasi balon udara menggunakan GPS (*Global Posistioning System*) dan kompas sehingga balon udara dapat melakukan navigasi secara otomatis menuju *waypoint*. Dengan sistem navigasi otomatis, balon udara dapat melakukan pemantauan dan eksplorasi tanpa bantuan operator. Sistem ini menggunakan sensor GPS berbasis GNSS (*Global Navigation Satellite System*) untuk mengetahui posisi, sensor kompas berbasis IMU (*Inertial Measurement Unit*) untuk mengetahui arah, dan kontrol logika *fuzzy* sebagai pengatur kecepatan motor kemudi untuk pergerakan balon. Pada metode ini, akurasi dari sensor GNSS berkisar antara 9,5 sampai 20 meter. Galat maksimum dari kompas dengan kompensasi kemiringan adalah 7%. Terjadi osilasi pada arah Utara dengan simpangan berkisar antara 50-90 derajat dan arah Barat dengan simpangan sebesar 20-50 derajat.

Kata kunci: Balon udara, GPS, Kompas, Navigasi, *Waypoint*

AIRSHIP NAVIGATION SYSTEM USING GPS AND FUZZY LOGIC CONTROL

Name	**: Dimas Arief Rahman Kurniawan**
Advisor	**: Dr. Muhammad Rivai, ST., MT.**
Co-Advisor	**: Rudy Dikairono, ST., MT.**

ABSTRACT

The development of Unmanned Aerial Vehicle (UAV) navigation system for the purposes of monitoring and exploration is increasing. Commonly used UAV type for this purpose is quadcopter and fixed wing aircraft. Quadcopter has a compact form so it can fly in a narrow and small room. Whereas the fixed-wing aircraft are often used for large area mapping. However, the use of both vehicle is less suitable for the purposes of monitoring and exploration. Quadcopter has short flying time and distance. Fixed wing aircraft can fly in long term and distances, but has high cruising speed. Alternative UAV option for monitoring and exploration is a blimp. Blimp is suitable for exploration in low speed and low altitude. Blimp use helium gas for lifting force, so that blimp can fly longer with greater distances. Therefore, this final project design and researched of the blimp navigation system using GPS (Global Posistioning System) and a compass so that blimp can navigate automatically to the waypoint. With automatic navigation system, a blimp can do monitoring and exploration without operator assistance. The system uses GPS sensor based on GNSS (Global Navigation Satellite System) to determine the position, the compass sensor based on IMU (Inertial Measurement Unit) to determine the direction, and fuzzy logic control as the steering motor speed control to move the balloon. In this method, the accuracy of GNSS sensors between from 9.5 to 20 meters. The maximum error of tilt-compensated compass is 7%. Oscillation occurs in the North direction with deviations ranging between 50-90 degrees and the West direction with deviation of 20-50 degrees.

Key words: Blimp, GPS, Compass, Navigation, Waypoint

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pengembangan sistem navigasi *Unmanned Aerial Vehicle* (UAV) untuk keperluan pemantauan dan eksplorasi semakin meningkat. Jenis UAV yang sering digunakan untuk keperluan ini adalah *quadcopter* dan pesawat *fixed wing*. *Quadcopter* memiliki bentuk yang ringkas sehingga dapat terbang di tempat yang sempit dan kecil. Pesawat *fixed wing* sering digunakan untuk pemetaan area yang luas karena strukturnya yang aerodinamis sehingga dapat terbang lebih lama daripada *quadcopter*.

Namun penggunaan kedua wahana tersebut kurang cocok untuk keperluan pemantauan dan eksplorasi. Q*uadcopter* memiliki jangka terbang dan jarak tempuh yang pendek karena menggunakan perputaran rotor sebagai gaya angkatnya sehingga boros dalam penggunaan daya. Pesawat *fixed wing* jangka terbang dan jarak tempuh yang panjang, namun mempunyai kecepatan terbang yang tinggi. Selain itu, pesawat *fixed wing* membutuhkan landasan pacu untuk lepas landas dan mendarat. Pesawat *fixed wing* juga memerlukan angin yang berlawanan arah agar pesawat dapat terangkat sehingga menjadikan pesawat ini harus tetap bergerak untuk dapat terbang.

Salah satu pilihan UAV alternatif untuk keperluan pemantauan dan eksplorasi adalah balon udara. Balon udara sangat cocok digunakan untuk aplikasi ekplorasi dalam kecepatan dan ketinggian yang rendah [1]. Gaya angkat balon udara menggunakan gas helium, sehingga balon udara dapat terbang lebih lama dan dengan jarak yang jauh. Balon udara juga tidak memerlukan landasan pacu untuk dapat terbang. Oleh karena itu pada tugas akhir ini merancang dan merealisasikan sistem navigasi balon udara menggunakan GPS (*Global Posistioning System*) dan kompas sehingga balon udara dapat melakukan navigasi secara otomatis menuju *waypoint*. Dengan sistem navigasi otomatis, balon udara dapat melakukan monitoring dan eksplorasi tanpa bantuan operator.

Harapan dari teknologi ini adalah balon udara dapat bergerak dengan sistem navigasi berbasis *waypoint* secara otomatis untuk untuk keperluan pemantauan dan eksplorasi.

1.2. Perumusan Masalah

Berdasarkan latar belakang di atas, dapat dirumuskan beberapa masalah, antara lain :

1. Bagaimana merancang sistem navigasi balon udara?
2. Bagaimana menavigasi balon udara?
3. Bagaimana mengatur motor kemudi untuk mengarahkan balon ke *waypoint*?
4. Bagaimana mengkompensasi pembacaan kompas terhadap gangguan kemiringan pada balon?

1.3. Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah :

1. Take off dan landing dilakukan secara manual.
2. Navigasi menuju koordinat *waypoint*.
3. *Waypoint* telah ditentukan.

1.4. Tujuan

Tujuan yang ingin dicapai dalam perancangan ini adalah:

1. Sistem navigasi balon udara menggunakan GPS dan kompas.
2. Sistem menavigasi balon menuju koordinat *waypoint*.
3. Pengaturan motor kemudi menggunakan kontrol logika *fuzzy*.
4. Kompensasi pembacaan kompas terhadap gangguan kemiringan menggunakan algoritma AHRS (*Attitude Heading Reference System*).

1.5. Metodologi

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut :

1. Studi Literatur

Studi literatur berguna untuk mencari informasi atau data mengenai balon udara, kontroler, atau sistem secara keseluruhan. Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan Tugas Akhir. Dasar teori ini dapat diambil dari buku-buku, jurnal, *proceeding*, dan artikel-artikel di internet dan forum-forum diskusi internet. Dengan adanya studi literatur, penelitian dapat dilakukan berdasarkan teori-teori yang telah ada sebelumnya.

2. Perancangan Sistem

Setelah mempelajari literatur yang ada, selanjutnya akan dilakukan perancangan sistem. Perancangan sistem terbagi sebagai berikut :

a. Perancangan Perangkat Keras

Sistem navigasi otomatis dari balon udara ini menggunakan metode navigasi berbasis *waypoint,* di mana metode ini didukung dengan penggunaan 2 sensor, yaitu kompas untuk mendeteksi derajat arah gerak balon udara (*heading*) dan sensor GNSS (*Global Navigation Satellite System*) untuk mendeteksi posisi balon udara. Untuk mengarahkan balon menuju *waypoint* digunakan 2 buah motor samping.

b. Perancangan Perangkat Lunak

Perancangan perangkat lunak meliputi proses pembacaan sensor kompas dan GNSS pada mikrokontroller, algoritma AHRS untuk mengkompensasi pembacaan kompas terhadap gangguan kemiringan balon, algoritma untuk menenentukan arah gerak balon menuju *waypoint*, dan sistem pengaturan kecepatan motor menggunakan kontrol logika *fuzzy*. Mikrokontroller pada balon akan terus mengolah data sensor untuk menentukan arah gerak balon udara.

3. Pengujian Sistem

Pengujian alat dilakukan untuk menentukan keandalan dari sistem yang telah dirancang. Pengujian dilakukan untuk melihat apakah *software* dan *hardware* dapat bekerja secara baik.

Untuk pengujian dilakukan dalam lima tahap. Pertama adalah pengujian berat maksimum balon. Kedua adalah pengujian keseimbangan balon. Ketiga adalah pengujian kompensasi pembacaan kompas terhadap gangguan kemiringan. Keempat adalah pengujian pembacaan posisi pada sensor GNSS. Dan terakhir adalah pengujian sistem navigasi untuk mengendalikan arah balon udara menuju *waypoint*.

4. Analisa

Analisa dilakukan terhadap hasil dari pengujian sehingga dapat ditentukan karakteristik dari *software* dan *hardware* yang telah dibuat. Apabila karakteristik dari *software* dan *hardware* pada sistem navigasi balon udara yang telah dibuat masih belum

sesuai, maka perlu dilakukan perancangan ulang pada sistem dan diuji kembali.

5. **Penyusunan Laporan Tugas Akhir**

   Tahap penulisan laporan tugas akhir adalah tahapan terakhir dari proses pengerjaan tugas akhir ini. Laporan tugas akhir berisi seluruh hal yang berkaitan dengan tugas akhir yang telah dikerjakan yaitu meliputi pendahuluan, teori penunjang, perancangan sistem, pengujian, dan penutup.

1.6. Sistematika Penulisan

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat terbagi menjadi lima bab dengan sistematika penulisan sebagai berikut :

- BAB I : PENDAHULUAN

  Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.

- BAB II : TEORI PENUNJANG

  Bab ini menjelaskan tentang teori penunjang dan literatur yang dibutuhkan dalam pengerjakan tugas akhir ini. Dasar teori yang menunjang meliputi teori dasar balon udara, kompensator apung, sensor kompas, AHRS (*Attitude Heading Reference System*), kuarternion, sudut Euler, sensor GNSS (*Global Navigation Satellite System*), metode pengukuran *geodistance*, metode pengukuran *geobearing*, dan logika *fuzzy*. Bagian ini memaparkan mengenai beberapa teori penunjang dan beberapa literatur yang berguna bagi pembuatan Tugas Akhir ini.

- BAB III : PERANCANGAN SISTEM

  Bab ini menjelaskan tentang perencanaan balon udara, sistem elektrik, mekanik, serta perangkat lunak. Bab ini juga berisi menjelaskan tentang prosedur pengujian yang dilakukan dalam penelitian.

- BAB IV : PENGUJIAN

  Pada bab ini akan menjelaskan hasil uji coba sistem beserta analisanya.

- BAB V : PENUTUP

  Bagian ini merupakan bagian akhir yang berisikan kesimpulan yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran untuk pengembangan lebih lanjut.

1.7. Relevansi dan Manfaat

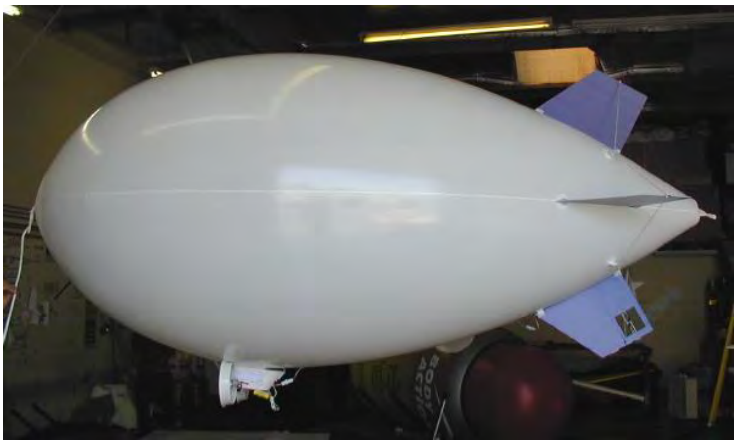
Hasil yang diharapkan dari tugas akhir ini diharapkan mampu menjadi referensi dalam pengembangan sistem navigasi pada wahana nirawak. Pengembangan lebih lanjut dari sistem ini adalah penambahan sistem navigasi berbasis inersia untuk melengkapi sistem navigasi menggunakan kompas dan GNSS sehingga sistem ini dapat menentukan posisi dengan lebih akurat.

Halaman ini sengaja dikosongkan

BAB II
TEORI PENUNJANG

Teori penunjang dalam bab ini menjelaskan tentang teori penunjang yang berhubungan dengan keseluruhan sistem yang akan dibuat pada tugas akhir ini.

2.1. Balon Udara



Gambar 2.1 Balon Udara [2]

Balon udara merupakan sebuah pesawat udara yang terbang dengan memanfaatkan gas angkat yang bersifat *aerostat* atau bersifat lebih ringan dari udara. Pada awal ditemukannya balon udara, gas angkat yang digunakan adalah hidrogen karena kapasitas angkat yang tinggi yaitu 1,15 kg/m^3. Karena hidrogen mudah terbakar, maka gas yang digunakan adalah helium yang memiliki gaya angkat 1 kg/m^3.

Ada 3 jenis balon udara, yaitu balon udara berangka, balon udara semi-rangka dan dan balon udara tanpa rangka. Balon udara tanpa rangka, adalah sebuah balon udara tanpa struktur kerangka internal. Tidak seperti balon udara semi-rangka (misalnya Zeppelin), balon udara tanpa rangka mengandalkan gaya angkat gas (biasanya helium, daripada hidrogen) di dalam penampung gas berupa balon untuk melayang di udara dan kekuatan gaya angkat gas itu sendiri mempertahankan bentuknya.

Balon udara yang digunakan pada tugas akhir ini adalah jenis balon udara tanpa rangka seperti terlihat pada gambar 2.1 yang menggunakan gas helium sebagai gas angkatnya. Untuk stabilisasi kemiringan balon, biasanya balon digantungkan beban, sehingga kemiringan balon dapat diatur sesuai keinginan. Selain itu beban tersebut juga digunakan sebagai pemberat balon agar tidak melayang terlalu tinggi. Beban yang digunakan tersebut bernama kompensator apung.

2.1.1. Kompensator Apung

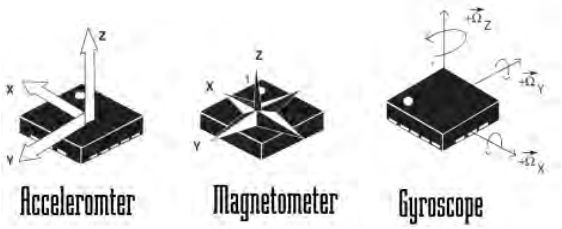
Kompensator apung pada balon udara sangat diperlukan karena daya apung pada saat balon udara terbang tidak konstan. Kompensasi ini juga berguna untuk mengatur ketinggian pada balon udara. Kompensator apung yang umum digunakan adalah menggunakan air dan gas yang memiliki kepadatan yang mendekati kepadatan udara.

2.2. Sensor Kompas

Kompas adalah alat navigasi untuk menentukan arah berdasarkan dengan medan magnet bumi. Kompas memberikan rujukan arah tertentu, sehingga sangat membantu dalam bidang navigasi. Kompas digital dapat dibuat dari sensor *inertial measurement unit* (IMU). Sensor IMU terdiri dari 3 komponen sensor yaitu akselerometer, giroskop, dan magnetometer. Magnetometer pada sensor IMU dapat mendeteksi medan magnet bumi sehingga dapat dijadikan acuan pada kompas digital.

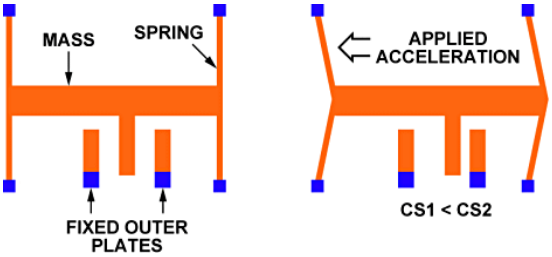
2.2.1. Sensor IMU (*Inertial Measurement Unit)*

IMU adalah singkatan dari *Inertial Measurement Unit*, adalah sebuah sensor yang digunakan untuk mengukur sudut *roll*, sudut *pitch* dan sudut *yaw*. Ketiga sudut ini menentukan orientasi dari sebuah pesawat udara pada saat terbang relatif terhadap bumi. Dalam mengukur sudut orientasi pesawat udara, IMU memerlukan tiga sensor pendukung yaitu magnetometer, akselerometer, giroskop (Gambar 2.2).



Gambar 2.2 Komponen sensor IMU [3]

2.2.2. Akselerometer



Gambar 2.3 MEMS Akselerometer [4]

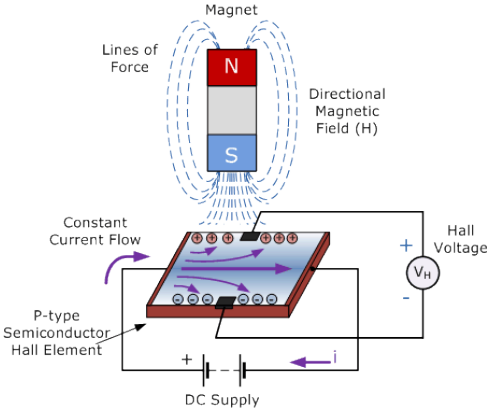
Akselerometer adalah sensor yang digunakan untuk mengukur percepatan suatu objek. Akselerometer dapat mengukur percepatan dinamis dan statis. Pengukuran percepatan dinamis adalah pengukuran percepatan pada objek yang bergerak (Gambar 2.3), sedangkan pengukuran percepatan statis adalah pengukuran percepatan gravitasi bumi. Akselerometer dapat digunakan untuk mengukur sudut kemiringan (*tilt*).

Data yang diperoleh modul sensor akselerometer adalah data mentah percepatan gravitasi yang direpresentasikan dalam *signed integer* 16 bit. Pada perancangan tugas akhir ini, skala penuh yang dipilih adalah 8g artinya data mentah sejumlah 65536 mewakili kecepatan sudut mulai 0g hingga 8g. Sensitifitas modul sensor saat skala penuh ini adalah 4 mg/digit.

2.2.3. Magnetometer

Magnetometer pada sensor IMU dapat mendeteksi medan magnet bumi sehingga dapat dijadikan acuan pada kompas digital. Sensor kompas digital dan sensor magnetometer pada hakikatnya memiliki prinsip kerja yang sama. Sensor kompas digital merupakan modul sensor magnetometer dengan keluaran berupa sudut yang menyatakan arah hadap. Sedangkan sensor magnetometer keluarannya berupa besar medan magnet bumi yang diukur dalam tiga sumbu yang dapat digunakan untuk menentukan sudut arah hadap dengan rumusan tertentu.

Data yang diperoleh modul sensor magnetometer adalah data mentah gaya medan magnet bumi yang direpresentasikan dalam *signed integer* 16 bit. Di perancangan, skala penuh yang dipilih adalah 8,1 gauss artinya data mentah sejumlah 65536 mewakili gaya medan magnet bumi mulai 0 gauss hingga 8,1 gauss.



Gambar 2.4 Magnetometer bertipe *Hall effect* [5]

Pada sensor magnetometer, untuk menghindari kesalahan pengukuran pada keadaan sensor miring, dapat digunakan perhitungan kompensasi kemiringan dengan menggunakan keluaran sensor akselerometer. Jenis sensor magnetometer yang digunakan pada tugas akhir ini adalah menggunakan *Hall Effect* (Gambar 2.4).

2.2.4. Giroskop

Giroskop digunakan untuk mengukur orientasi berdasarkan prinsip momentum sudut. Giroskop mengukur kecepatan sudut kerangka acuan inersia. Sudut orientasi berupa gerak *roll*, *pitch*, dan *yaw* didapatkan dengan mengintegrasikan kecepatan sudut. Sehingga hasil giroskop adalah percepatan sudut.

Giroskop berbeda dengan akselerometer dan kompas. Akselerometer mengukur gerakan linier acuan gravitasi. Akselerometer dapat memberikan pengukuran sudut kemiringan (*tilt*) akurat ketika suatu sistem dalam keadaan diam (statis). Saat sistem berotasi atau bergerak, akselerometer tidak dapat mengikuti pergerakan yang cepat dikarenakan responnya lambat dan memiliki noise, sehingga tidak dapat digunakan untuk pengukuran sudut orientasi dalam kendaraan udara.

Sedangkan kompas mengukur gerakan linier dengan acuan medan magnet bumi. Kompas dapat mengukur gerak *yaw* atau arah mata angin namun tidak dapat mengukur gerak *roll* dan *pitch*.

Keluaran giroskop berupa data kecepatan sudut. Kecepatan sudut adalah besaran vektor yang menyatakan frekuensi sudut suatu benda

terhadap sumbu putarnya. Satuan untuk kecepatan sudut adalah radian per detik.

2.3. AHRS (*Attitude Heading Reference System*)

Attitude Heading Reference System terdiri dari sensor pada tiga sumbu yang dapat menunjukkan informasi orientasi pesawat termasuk *roll*, *pitch*, dan *yaw*. AHRS dirancang untuk menggantikan instrumen penerbangan yang berupa giroskop mekanis tradisional dan memberikan kehandalan dan akurasi.

AHRS terdiri dari sensor *gyroscope*, *accelerometer*, *magnetometer* pada ketiga sumbu. Perbedaan utama antara IMU dan AHRS adalah pada AHRS ditambahkan pemroses informasi *attitude* dan *heading* dalam satu perangkat dibandingkan IMU yang memberikan data sensor pada perangkat tambahan untuk menghitung *attitude* dan *heading*. dapat Keluaran dari AHRS adalah berupa sudut Euler. IMU juga dapat digunakan sebagai perangkat AHRS dengan tambahan pemroses data mentah sensor.

Estimasi non linear seperti Kalman Filter dapat digunakan untuk menghitung informasi AHRS dari sensor IMU. Namun perhitungan kompleks pada Kalman filter sangat sulit diterapkan pada mikrokontroller karena keterbatasan pemrosesan data.

Pada tugas akhir ini digunakan filter kuarternion untuk mendapatkan data AHRS dan kemudian dikonversi menjadi sudut Euler. Perhitungan pada filter kuarternion lebih sederhana daripada Kalman filter sehingga pemrosesan pada mikrokontroller dapat dilakukan dengan lebih cepat.

2.3.1. Kuarternion

Dalam matematika, Kuaternion merupakan perluasan dari bilangan-bilangan kompleks yang tidak komutatif, dan diterapkan dalam mekanika tiga dimensi. Kuaternion ditemukan oleh ahli matematika dan astronomi Inggris, William Rowan Hamilton, yang menurunkan aritmatika kompleks ke kuaternion [6].

Sebagai himpunan, kuaternion berlambang **H** (dinotasikan sesuai orang yang menemukannya Hamilton), sama dengan $\mathbf{R}^4$ yang merupakan ruang vektor bilangan riil empat dimensi. **H** memiliki tiga macam operasi: pertambahan, perkalian skalar dan perkalian kuaternion. Elemen-elemen kuaternion ditandakan sebagai 1, i, j dan k (i, j dan k adalah komponen imaginer), dan dapat ditulis sebagai kombinasi linear, $a + bi + cj + dk$ (a, b, c, dan d adalah bilangan riil).

Tabel 2.1 Aturan Kuarternion

x	l	i	j	K
l	l	i	j	k
i	i	-l	k	-j
j	j	-k	-l	i
k	k	j	-i	-l

Konsep dari kuaternion adalah membagi sumbu kordinat yang semula 3 dimensi menjadi 4 dimensi.Kuaternion adalah matrik yang yang terdiri atas vektor dan skalar. Kenapa di sebut kuaternion karena pada dasarnya kuarternion mempunyai 4 elemen "q". Terdiri q_0 – q_3, dimana q_0 = indikasi dimensi dari arah vector, q_1 = mewakili vektor sumbu x, q_2 =mewakili vektor sumbu y, q_3 = mewakili vektor sumbu z. dapat dilihat pada tabel 2.1 aturan perkalian bilangan kompleks pada kuaternion.

Keluaran dari perhitungan kuarternion adalah berupa satuan kuarternion yaitu :

$$\boldsymbol{q} = [q_0\ q_1\ q_2\ q_3]^T \tag{2.1}$$

Kuarternion dapat diturunkan menjadi rotasi pada sumbu dengan persamaan berikut ini :

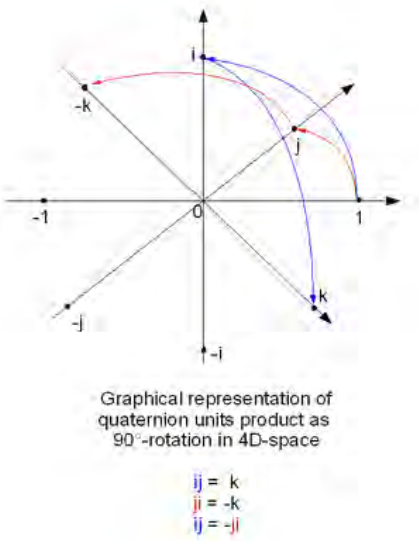
$$\boldsymbol{q}_0 = \cos\left(\frac{\alpha}{2}\right) \tag{2.2}$$

$$\boldsymbol{q}_1 = \sin\left(\frac{\alpha}{2}\right) \times \cos(\beta_x) \tag{2.3}$$

$$\boldsymbol{q}_2 = \sin\left(\frac{\alpha}{2}\right) \times \cos(\beta_y) \tag{2.4}$$

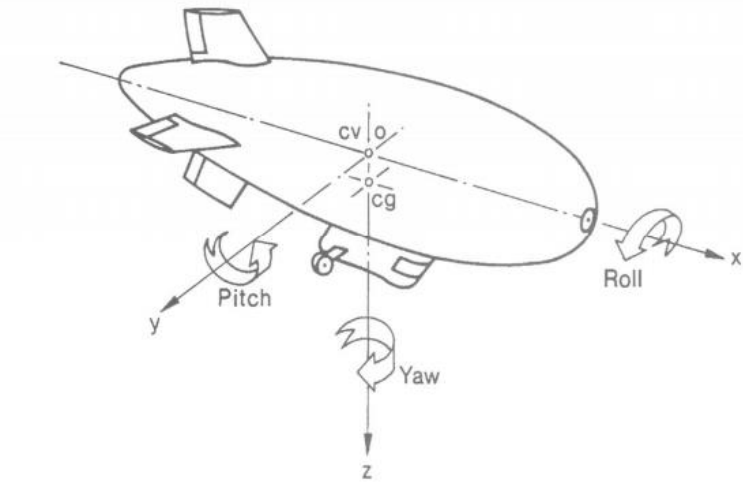
$$\boldsymbol{q}_3 = \sin\left(\frac{\alpha}{2}\right) \times \cos(\beta_z) \tag{2.5}$$

di mana α adalah sudut rotasi sederhana (nilai dalam radian dari sudut rotasi) dan cos (βx), cos (βy) dan cos (βz) adalah "arah cosinus" pada sumbu rotasi (Teorema Euler). Representasi kuarternion dalam rung 4 dimensi dapat terlihat pada gambar 2.5

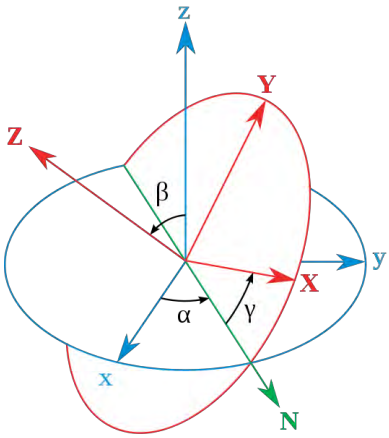


Gambar 2.5 Representasi Kuarternion [7]

2.3.2. Sudut Euler



Gambar 2.6 Rotasi sudut Euler [8]



Gambar 2.7 Pendekatan kuarternion dengan sudut Euler [9]

Sudut Euler merepresentasikan orientasi tiga dimensi dari suatu objek menggunakan kombinasi tiga rotasi sumbu yang berbeda. Informasi orientasi menggunakan sudut Euler lebih sederhana daripada kuarternion. Rotasi sudut Euler pada sumbu x, y, dan z dilambangkan dengan nama *roll* (Φ), *pitch* (θ), dan *yaw* (ψ) seperti terlihat pada gambar 2.6.

Berikut ini adalah persamaan untuk mengkonversi satuan kuarternion menjadi sudut Euler sesuai pendekatan kuarternion dengan sudut Euler pada gambar 2.7 :

$$\begin{bmatrix} \Phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}\big(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)\big) \\ \arcsin\big(2(q_0q_1 - q_2q_3)\big) \\ \text{atan2}\big(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)\big) \end{bmatrix} \tag{2.6}$$

Ada nilai pada saat orientasi sensor tidak dapat direpresentasikan menggunakan sudut Euler. Orientasi tersebut terjadi pada saat posisi *pitch* 90 derajat, sehingga *yaw* dan *roll* menjadi satu sumbu. Kondisi tersebut dinamakan *Gimbal Lock* (Gambar 2.8).

Sensor orientasi atau AHRS yang menggunakan sudut Euler akan selalu gagal untuk menghasilkan informasi ketika sudut *pitch* mendekati 90 derajat. Ini adalah masalah mendasar sudut Euler dan hanya dapat diselesaikan dengan beralih ke metode representasi yang berbeda.

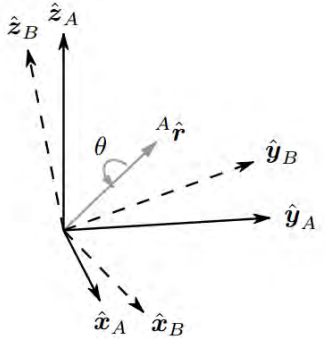
Gambar 2.8 Kondisi *Gimbal Lock* [10]

2.3.3. Konsep Magdwick AHRS

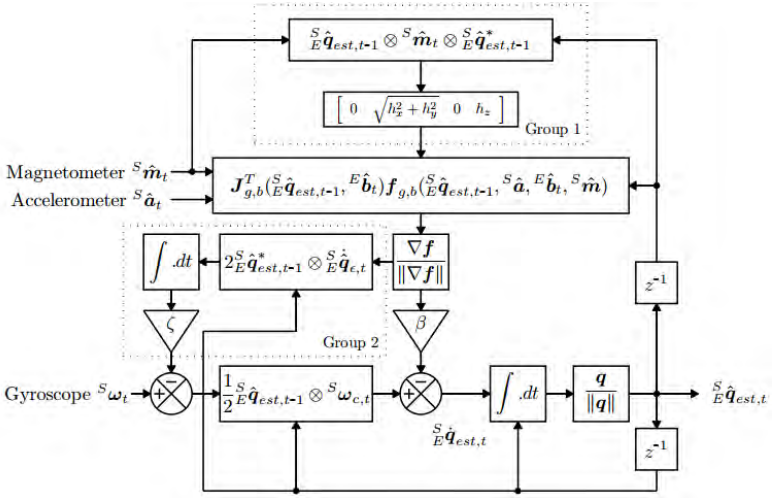
Konsep dasar dari *Magwick AHRS* adalah mencari antara relatif *frame* atau yang biasa disebut perpindahan *frame*. Contoh dapat dilihat pada gambar 2.9, *frame* A relatif terhadap *frame* B. Dimana nilai perpindahan *frame* A ke *frame* B dapat direpresentasikan dalam bentuk kuaternion seperti persamaan 2.7.

$$ {}^{A}_{B}\hat{q} = [q0\ q1\ q2\ q3] = \left[\cos\frac{\theta}{2} - rx\sin\frac{\theta}{2} - ry\sin\frac{\theta}{2} - rz\sin\frac{\theta}{2}\right] \quad (2.7) $$

$$ {}^{A}_{B}\hat{q} *= {}^{B}_{A}\hat{q} = [q0 - q1 - q2 - q3] \quad (2.8) $$



Gambar 2.9 Frame Kuaternion [11]



Gambar 2.10 Diagram Blok Algoritma AHRS Madgwick [11]

Untuk mencari nilai matrik yang baru, kedua matrik tersebut dikalikan dengan aturan *cross product* dimana dengan mengunakan aturan Hamilton. Seperti persamaan 2.9.

$$a \otimes b = [a0\ a1\ a2\ a3] \otimes [b0\ b1\ b2\ b3]$$

$$= \begin{bmatrix} a_0b_0 & -a_1b_1 & -a_2b_2 & -a_3b_3 \\ a_0b_1 & +a_1b_0 & +a_2b_3 & -a_3b_2 \\ a_0b_2 & -a_1b_3 & +a_2b_0 & +a_3b_1 \\ a_0b_3 & +a_1b_2 & -a_2b_1 & +a_3b_0 \end{bmatrix}^T \quad (2.9)$$

Pada gambar 2.10 dijelaskan alur dari metode madgwick. Untuk mengetahui representasi pergeserah *frame* berdasarkan sumbu garis normal gravitasi bumi menggunakan persamaan *gradient decent*. Dimana alogaritma *gradient decent* mempunyai fungsi seperti persamaan (2.10). Dengan demikian apabila dicross product ketiga kuaternion tersebut akan menghasilkan matrik baru (2.14).

$$f({}^S_E\hat{q}, E_{\hat{d}}, S_{\hat{s}}) = {}^S_E\hat{q} *\otimes E_{\hat{d}} \otimes {}^S_E\hat{q} - S_{\hat{s}}\ 2.26 \quad (2.10)$$

$${}^S_E\hat{q} = [q1 - q2 - q3 - q4] \quad (2.11)$$

$$E_{\hat{g}} = [0\ \ 0\ \ 0\ \ 1] \quad (2.12)$$

$$S_{\hat{a}} = [0\ ax\ ay\ az] \tag{2.13}$$

$$\mathrm{f_g}(\hat{{}_E^S q}, S_{\hat{a}}) = \begin{bmatrix} 2(q2q4 - q1q3) - ax \\ 2(q1q2 + q3q4) - ay \\ 2(\frac{1}{2} - {q_2}^2 - {q_3}^2) - az \end{bmatrix} \tag{2.14}$$

Setelah proses untuk mencari nilai f , kemudian dicari hasil dari matrik *jacobi* sesuai degan persamaan (2.15).

$$\mathrm{J_g}(\hat{{}_E^S q}) = \begin{bmatrix} -2q3 & 2q4 & -2q1 & 2q2 \\ 2q2 & 2q1 & 2q4 & 2q3 \\ 0 & -4q2 & -4q3 & 0 \end{bmatrix} \tag{2.15}$$

Kemudian setelah kedua matrik *f* dan *J* telah didapatkan hasil.Untuk mencari nilai dari matrik *gradien decent* adalah sesuai dengan persamaan (2.16).

$$\nabla \mathrm{f} = \mathrm{j^T} \times \mathrm{f} \tag{2.16}$$

Untuk mencari nilai estimasi dari kuaternion, kuaternion pada giroskop dikurangkan dengan hasil kuaternion pada proses alogaritma *gradien decent* yang sudah dibagi dengan nilai normalisasi dan dikalikan dengan nilai pembobotan (*beta*), sesuai dengan persamaan (2.17).

$$\dot{{}_E^S \mathrm{q}}\mathrm{est}, \mathrm{t} = \dot{{}_E^S \mathrm{q}}\omega, \mathrm{t} - \beta \frac{\nabla \mathrm{f}}{\|\nabla \mathrm{f}\|} \tag{2.17}$$

Untuk mencari nilai β atau pembobotan dapat menggunakan persamaan(2.18).

$$\beta = \left\| \frac{1}{2} \hat{q} \otimes [0\ \bar{\omega}_{max}\ \bar{\omega}_{max}\ \bar{\omega}_{max}] \right\| \sqrt{\frac{3}{4}}\, \bar{\omega}_{max} \tag{2.18}$$

2.4. Sensor GNSS (*Global Navigation Satellite System*)

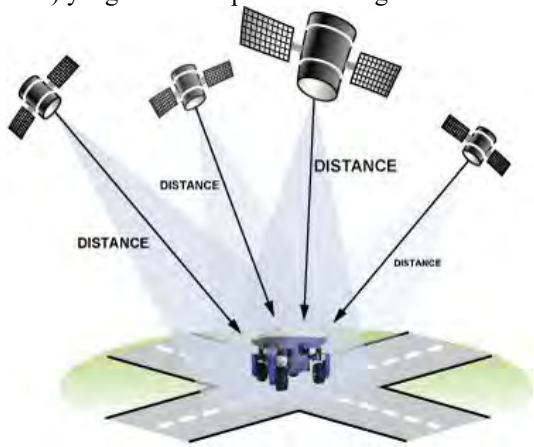
GNSS (*Global Navigation Satellite System*) merupakan suatu metode pengoperasian satelit yang terdiri dari gabungan beberapa sistem satelit navigasi seperti GPS (Amerika), GLONASS (Rusia), Galileo (Uni-Eropa), dan Beidou (Cina). GNSS disediakan untuk kepentingan sipil maupun militer di seluruh dunia. GNSS menyediakan informasi posisi, ketinggian, kecepatan, dan waktu dari *receiver*, sehingga memungkinkan pengguna untuk mengetahui lokasi tepat dimanapun di permukaan bumi [12].

Pada sistem navigasi satelit GPS, hanya satelit-satelit GPS saja yang dapat menyediakan informasi mengenai posisi, ketinggian, kecepatan, dan waktu dari *receiver*. Sedangkan pada GNSS, ia akan menggabungkan sisten navigasi satelit yang ada (misalnya GPS dan GLONASS) sehingga sinyal satelit yang diterima oleh suatu *receiver* semakin banyak. Semakin banyak sinyal satelit yang ditangkap oleh *receiver*, semakin banyak pula data yang didapat, sehingga mempengaruhi tingkat ketelitian informasi yang dibutuhkan.

GNSS dapat mendeteksi lokasi benda bergerak, sehingga dengan GNSS akan dapat diketahui posisi, arah, dan kecepatan benda secara langsung ataupun tertunda, bahkan posisi dapat diketahui secara tiga dimensi. Data pada GNSS dapat digunakan untuk mengetahui pergerakan atau rute suatu benda.

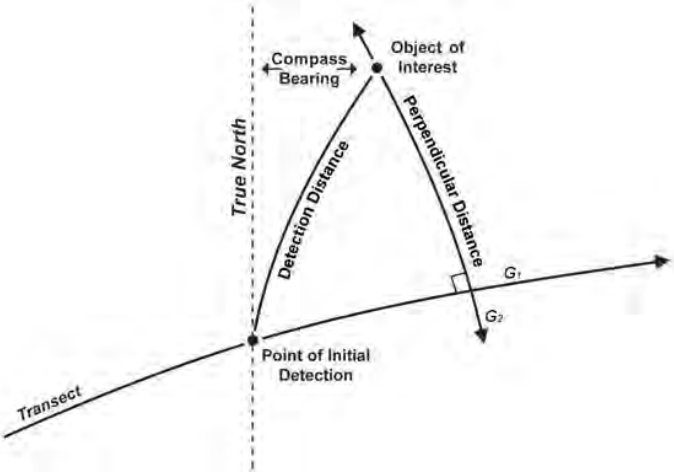
Untuk mengetahui lokasi suatu benda, receiver GPS memerlukan dua data, yaitu data waktu dan posisi satelit GPS. Satelit GPS secara terus menerus memancarkan sinyal yang berisi data kapan sinyal tersebut dikirim seperti pada gambar 2.11 dan lokasi satelit yang mengirim sinyal tersebut. Dengan memperoleh data kapan sinyal dikirim satelit GPS dan diterima oleh *receiver*, *receiver* GPS dapat mengetahui jarak *receiver* dengan satelit [13]. Data dari tiga satelit dapat diketahui lokasi *receiver* dan dengan data dari 4 satelit dapat diketahui lokasi receiver secara 3D, yaitu lokasi *longitude* (lintang), *latitude* (bujur), dan *altitude* (ketinggian).

Sejak April 2013, hanya sistem GNSS NAVSTAR-GPS (USA) dan GLONASS (Rusia) yang telah beroperasi secara global.



Gambar 2.11 Mendeteksi posisi dengan GNSS [13]

2.5. Metode Pengukuran *Geodistance*



Gambar 2.12 Pengukuran jarak pada permukaan bola [14]

Geodistance adalah jarak antara dua titik koordinat pada permukaan bumi. Gambar 2.12 diatas merupakan ilustrasi dari pengukuran jarak pada permukaan bola. Permukaan bumi tidak planar/datar melainkan berbentuk bola, sehingga pengukuran jarak antara dua titik koordinat tidak dapat dilakukan dengan metode planar. Untuk mengukur jarak pada permukaan elips bumi digunakan persamaan *Haversine*. Persamaan tersebut menghitung jarak terpendek antara dua titik koordinat pada permukaan bumi. Persamaan *Haversine* dinyatakan melalui persamaan berikut :

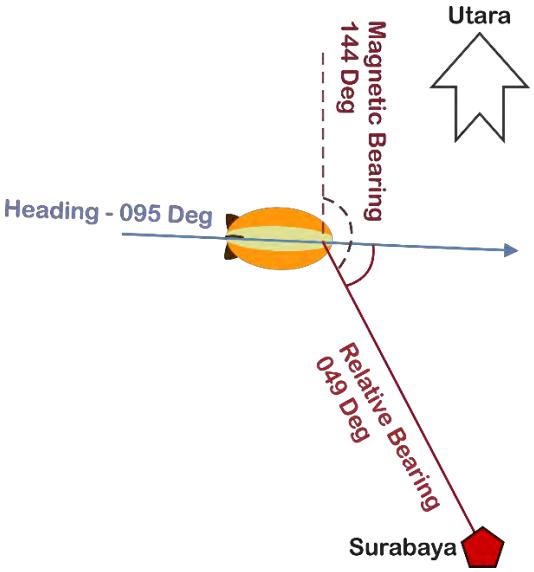
$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + cos\,\varphi_1 \cdot cos\,\varphi_2 \cdot sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{2.19}$$

$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{1-a}) \tag{2.20}$$

$$d = R \cdot c \tag{2.21}$$

Dimana :
$\varphi = Latitude\ dalam\ radian$
$\lambda = Longitude\ dalam\ radian$
$c = Jarak\ angular$
$R = Jari - jari\ bumi\ (6371\ km)$
$d = Jarak\ pada\ permukaan\ bumi$

2.6. Metode Pengukuran *Geobearing*



Gambar 2.13 *Geobearing*

Secara umum, *relative bearing* adalah sudut antara arah hadap objek (*heading)* dengan target. Sedangkan *magnetic bearing* adalah sudut antara arah utara dengan target. *Relative bearing* dapat dihitung dengan cara mengurangkan *magnetic bearing* dengan *heading*. Pada gambar 2.13, *heading* yang terukur adalah sebesar 95 derajat dan *magnetic bearing* yang terukur adalah 144 derajat, maka *relative bearing* adalah sebesar 49 derajat.

Persamaan dibawah ini adalah persamaan yang diperlukan untuk menghitung *magnetic bearing* jika ditarik garis lurus dari posisi sekarang menuju target :

$$\theta = atan2(\sin\Delta\lambda \cdot \cos\varphi_2 , \cos\varphi_1 \cdot \sin\varphi_2 - \sin\varphi_1 \cdot \cos\varphi_2 \cdot \cos\Delta\lambda) \quad (2.22)$$

Dimana :

$\varphi = Latitude\ dalam\ radian$

$\lambda = Longitude\ dalam\ radian$

$\theta = Bearing$

Karena atan2 menghasilkan nilai dalam kisaran -π sampai +π (-180° sampai +180 °), untuk menormalisasikan hasil *bearing* (kisaran 0° sampai 360 °) dengan mengkonversi ke derajat dan kemudian menggunakan persamaan (θ + 360)% 360, di mana % adalah (*floating point*) modulo.

Untuk *bearing* akhir, secara sederhana hanya mengambil *bearing* awal dari titik akhir ke titik awal dan melakukan *inverse* pada nilai tersebut (menggunakan θ = (θ + 180)% 360).

2.7. Logika *Fuzzy*

Konsep tentang logika *fuzzy* diperkenalkan oleh Prof. Lotfi Astor Zadeh pada 1962. Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah, yang cocok untuk diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, *embedded system*, jaringan PC, *workstation* berbasis akuisisi data, dan sistem kontrol. Metodologi ini dapat diterapkan pada perangkat keras, perangkat lunak, atau kombinasi keduanya. Dalam logika klasik dinyatakan bahwa segala sesuatu bersifat biner, yang artinya adalah hanya mempunyai dua kemungkinan, "Ya atau Tidak", "Benar atau Salah", "Baik atau Buruk", dan lain-lain. Oleh karena itu, semua ini dapat mempunyai nilai keanggotaan 0 atau 1. Akan tetapi, dalam logika *fuzzy* kemungkinan nilai keanggotaan berada diantara 0 dan 1. Artinya,bisa saja suatu keadaan mempunyai dua nilai "Ya dan Tidak", "Benar dan Salah", "Baik dan Buruk" secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya [15].

2.8.1. Operasi Himpunan *Fuzzy*

Operasi himpunan *fuzzy* diperlukan untuk proses inferensi atau penalaran. Dalam hal ini yang dioperasikan adalah derajat keanggotaanya. Derajat keanggotaan sebagai hasil dari operasi dua buah himpunan *fuzzy* disebut sebagai fire strength atau α-predikat [15].

Ada beberapa hal yang perlu diketahui dalam memahami sistem *fuzzy*, yaitu :

1. Variabel *fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem *fuzzy*.
2. Himpunan *fuzzy* merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *fuzzy*.
3. Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *fuzzy*. Semesta pembicaraan merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta

pembicaraan dapat berupa bilangan positif maupun negatif. Adakalanya nilai semesta pembicaraan ini tidak dibatasi batas atasnya.

4. Domain himpunan *fuzzy* adalah keseluruhan nilai yang diizinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*. Seperti halnya semesta pembicaraan, domain merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif maupun negatif [16] [17].

2.8.2. Fungsi Keanggotaan

Fungsi keanggotaan adalah grafik yang mewakili besar dari derajat keanggotaan masing-masing variabel input yang berada dalam interval antara 0 dan 1. Derajat keanggotaan sebuah variabel x dilambangkan dengan simbol μ(x). Rule- rule menggunakan nilai keanggotaan sebagai faktor bobot untuk menentukan pengaruhnya pada saat melakukan inferensi untuk menarik kesimpulan [15].

Ada beberapa fungsi yang bisa digunakan antara lain :

1. Representasi Linear, pada representasi linear pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Ada dua keadaan *fuzzy* yang linear yaitu representasi linear naik dan representasi linear turun.
2. Representasi Kurva Segitiga, Kurva segitiga pada dasarnya merupakan gabungan antara dua garis linear.
3. Representasi Kurva Trapesium, Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1.
4. Representasi Kurva Bentuk Bahu, Daerah yang terletak di tengah – tengah suatu variabel yang dipresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun. Tetapi terkadang salah satu sisi dari variabel tersebut tidak mengalami perubahan.
5. Representasi Kurva-S, Kurva Pertumbuhan dan Penyusutan merupakan kurva-S atau *sigmoid* yang berhubungan dengan kenaikan dan penurunan permukaan secara tak linear.
6. Representasi Kurva Bentuk Lonceng (*Bell Curve*), Untuk mempresentasikan bilangan *fuzzy*, biasanya digunakan kurva berbentuk lonceng. Kurva berbentuk lonceng ini terbagi atas tiga

kelas, yaitu kurva PI, kurva beta, dan kurva Gauss. Perbedaan ketiga kurva ini terletak pada gradientnya [17].

2.8.3. Logika *Fuzzy* Mamdani

Metode Mamdani paling sering digunakan dalam aplikasi-aplikasi karena strukturnya yang sederhana, yaitu menggunakan operasi MIN-MAX atau MAX-PRODUCT. Untuk mendapatkan output, diperlukan empat tahapan berikut [15] :

1. Pembentukan himpunan *fuzzy*. Pada proses fuzzifikasi langkah yang pertama adalah menentukan variable *fuzzy* dan himpunan fuzzinya. Kemudian tentukan derajat kesepadanan (degree of match) antara data masukan *fuzzy* dengan himpunan *fuzzy* yang telah didefenisikan untuk setiap variabel masukan sistem dari setiap aturan *fuzzy*. Pada metode mamdani, baik variabel input maupun variabel output dibagi menjadi satu atau lebih himpunan *fuzzy*.
2. Aplikasi fungsi implikasi pada metode mamdani. Fungsi implikasi yang digunakan adalah min. Lakukan implikasi *fuzzy* berdasar pada kuat penyulutan dan himpunan *fuzzy* terdefinisi untuk setiap variabel keluaran di dalam bagian konsekuensi dari setiap aturan. Hasil implikasi *fuzzy* dari setiap aturan ini kemudian digabungkan untuk menghasilkan keluaran infrensi *fuzzy* [16].
3. Komposisi Aturan. Tidak seperti penalaran monoton, apabila sistem terdiri dari beberapa aturan, maka infrensi diperoleh dari kumpulan dan korelasi antar aturan. Ada 3 metode yang digunakan dalam melakukan inferensi sistem *fuzzy*, yaitu: max, additive dan probabilistik OR.
4. Penegasan (*defuzzy*). Input dari proses defuzzifikasi adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan output yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut.

$$Z^* = \frac{\int u(z)zdz}{\int u(z)dz} \tag{2.23}$$

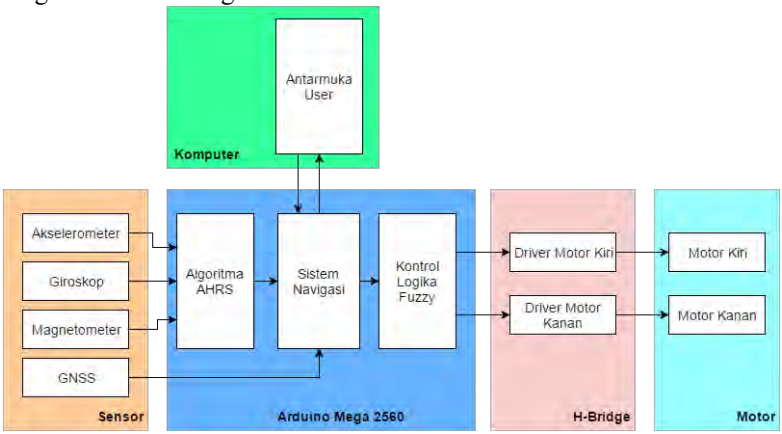
Halaman ini sengaja dikosongkan

BAB III
PERANCANGAN SISTEM

Pada bab ini akan dijelaskan perancangan sistem mulai dari perancangan perangkat keras hingga perancangan perangkat lunak. Perancangan perangkat keras meliputi perancangan mekanik, perancangan komponen elektrik, dan perancangan posisi gondola dan motor kemudi balon udara. Perancangan perangkat lunak meliputi proses akuisisi data sensor IMU dan GNSS, perancangan algoritma AHRS, perancangan sistem navigasi, dan perancangan kontrol logika *fuzzy*.

3.1. Diagram Blok Sistem

Pada tugas akhir ini, sensor GNSS dan kompas digunakan untuk mengetahui posisi balon dan arah hadap balon terhadap *waypoint*. Informasi tersebut kemudian dimasukkan pada mikrokontroller sehingga data dapat diproses pada kontrol logika *fuzzy* untuk menggerakkan motor balon. Motor tersebut akan mengarahkan balon ke koordinat *waypoint*. Pada sistem ini, *waypoint* dapat diinputkan dari antarmuka user dengan mikrokontroller menggunakan komunikasi serial. Berikut ini adalah diagram blok rancangan dasar sistem :

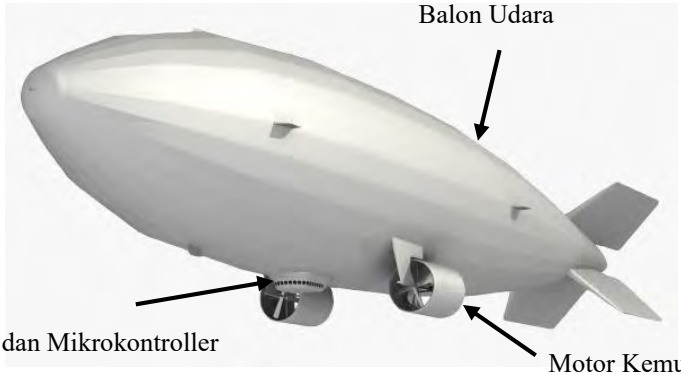


Gambar 3.1 Blok Diagram Rancangan Dasar Sistem

3.2. Perancangan Perangkat Keras

Pada tugas akhir ini, perangkat keras yang akan digunakan untuk membangun sistem ini adalah balon udara, rangkaian suplai daya, sensor IMU, sensor GNSS, Arduino Mega, *driver* motor, dan motor DC.

3.2.1. Balon Udara

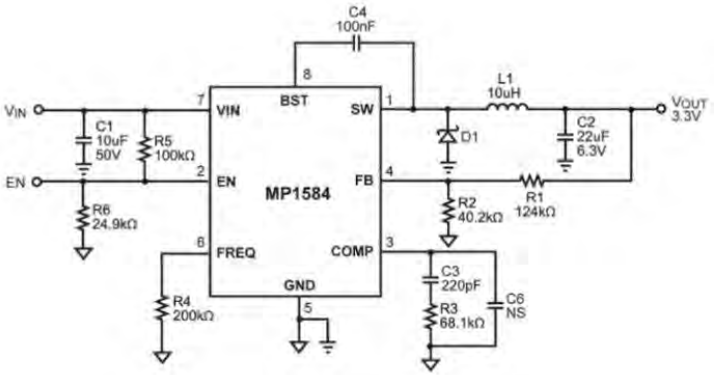


Gambar 3.2 Balon udara

Balon udara yang akan digunakan adalah berjenis *blinp*. *Blimp* memiliki bentuk yang aerodinamis sehingga *drag* dari blimp tidak terlalu besar. Hal tersebut berpengaruh pada pemakaian suplai daya, semakin rendah *drag*, maka semakin sedikit energi yang digunakan, maka balon udara jenis *blimp* dapat terbang dengan jarak yang jauh.

3.2.2. Rangkaian Suplai Daya

Suplai daya adalah perangkat elektronika yang mensuplai sumber listrik ke perangkat elektronika lainnya. Dalam suatu rangkaian suplai daya terdapat sebuah regulator tegangan dimana digunakan untuk menurunkan tegangan dari satu level tertentu ke level yang diinginkan. Dalam tugas akhir ini menggunakan sebuah regulator tegangan berupa modul MP1584 yang mampu meregulasi tegangan dari rentang 4-28 V_{input} menjadi 0,8-25 V_{output} dan mampu menyuplai suatu beban sampai batas arus 3 A. Besaran tegangan yang dibutuhkan ialah 7 V untuk suplai daya mikrokontroler, sensor, dan *driver* motor.



Gambar 3.3 Skematik *Buck Converter*

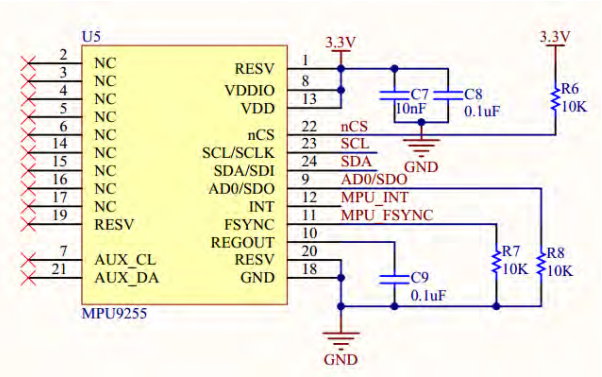
Pada gambar 3.3 kapasitor berfungsi sebagai pengaman pada tegangan input maupun pada output agar tegangan yang akan masuk maupun yang akan dikeluarkan tidak terinterferensi oleh noise akibat adanya loncatan arus. Untuk mendapatkan keluaran tegangan yang dinginkan menggunakan perbandingan besar R yang berbeda-beda. Berikut perhitungan rumus untuk menentukan besar perbandingan niali resistor:

$$Vout = \frac{Vfb(R1+R2)}{R2} \tag{3.1}$$

Pada beberapa aplikasi nilai R2 dijadikan 40,2 KOhm sehingga yang perlu dicari adalah besar nilai R1. Untuk nilai tegangan 7 V nilai R1 adalah 13,7 KOhm dan nilai R2 adalah 40,2 KOhm.

3.2.3. Sensor *Inertial Measurement Unit* (IMU)

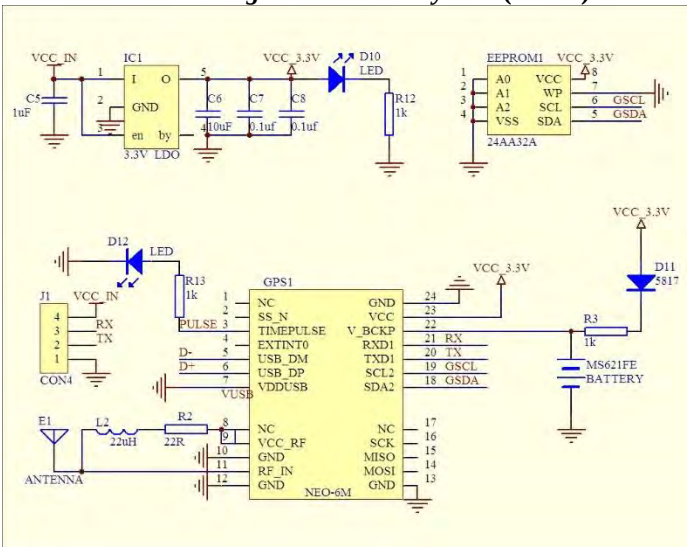
Inertial Measurement Unit (IMU) adalah sebuah sensor yang menggabungkan sensor giroskop, akselerometer dan magnetometer dalam sebuah devais. Sensor IMU yang digunakan pada tugas akhir ini adalah *Waveshare* 10 DoF IMU yang terdiri dari sebuah sensor IMU MPU 9255 dan sebuah barometer BMP180. Tegangan kerja sensor berada pada rentang 3,3 - 5 V.



Gambar 3.4 Skematik Dasar MPU9255

MPU 9255 adalah sensor IMU yang terdiri dari akselerometer dan giroskop buatan *Invensys*, serta magnetometer buatan *Asahi Kasei Microdevices* dalam satu chip. Sedangkan BMP180 adalah sensor barometer buatan *Bosch*. Antarmuka sensor ini menggunakan I2C.

3.2.4. Sensor *Global Navigation Satellite System* (GNSS)

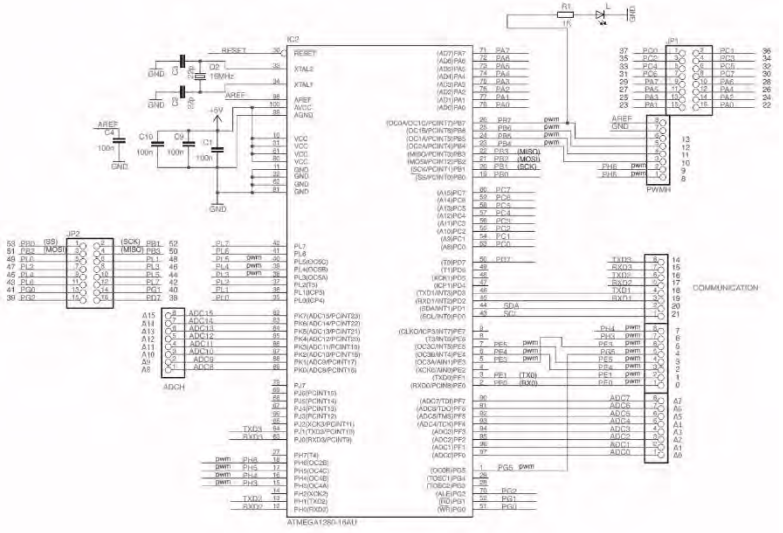


Gambar 3.5 Skematik Dasar Ublox M8N

Sensor GNSS yang digunakan pada tugas akhir ini adalah GY-GPSV3 yang terdiri dari sebuah Ublox Neo M8N. Sensor ini mampu menerima sinyal satelit dari sistem NAVSTAR-GPS milik Amerika dan GLONASS milik Rusia. Antarmuka sensor ini menggunakan UART. Sensor tersebut menggunakan antarmuka UART dengan tegangan kerja 3.6-5V.

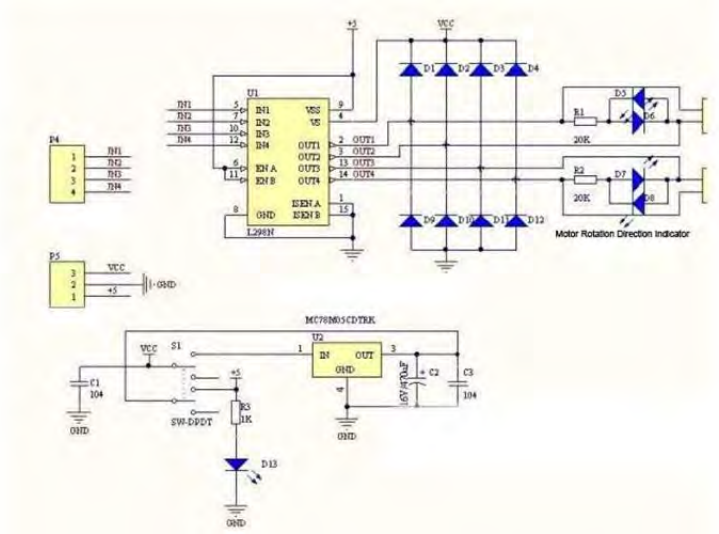
3.2.5. Arduino Mega

Processing unit yang digunakan pada tugas akhir ini sebagai pemroses data sistem navigasi adalah Arduino Mega 2560. Arduino Mega 2560 merupakan sebuah board mikrokontroler berbasis ATMega2560. Modul ini memiliki 54 digital input/output dimana 15 digunakan untuk PWM output dan 16 digunakan sebagai analog input, 4 port serial, 16 MHz osilator Kristal, ICISP Header, dan tombol reset. Arduino Mega 2560 memiliki flash memory sebesar 256KB. Arduino Mega 2560 tidak memerlukan *flash* program external karena di dalam chip mikrokontroler Arduino telah diprogram dengan *bootloader* yang membuat proses upload program yang dibuat menjadi lebih sederhana dan cepat.



Gambar 3.6 Skematik Arduino Mega 2560

3.2.6. *Driver* Motor



Gambar 3.7 Skematik *Driver* Motor L298N

Driver motor yang digunakan pada tugas akhir ini adalah sebuah L298N untuk menggerakkan motor kemudi. Tegangan suplai yang digunakan disesuaikan dengan tegangan kerja motor yang digunakan, yang dalam tugas akhir ini digunakan sebesar 7 Volt. Terdapat 3 input, yaitu IN1, IN2, ENA. IN1 dan IN2 merupakan pengatur arah gerak motor yang menentukan kutub positif dan negatif dari OUTA. Sedangkan ENA merupakan input tegangan DC antara 0-5 Volt yang berfungsi untuk mengatur kecepatan motor. Di pin ENA juga bisa diinputkan sinyal PWM.

Pengaturan kecepatan kedua motor dilakukan dengan cara pengontrolan lama pulsa aktif (mode PWM – *Pulse width Modulation*) yang dikirimkan ke rangkaian *driver* motor oleh mikrokontroller. *Duty cycle* PWM yang dikirimkan menentukan kecepatan putar motor DC.

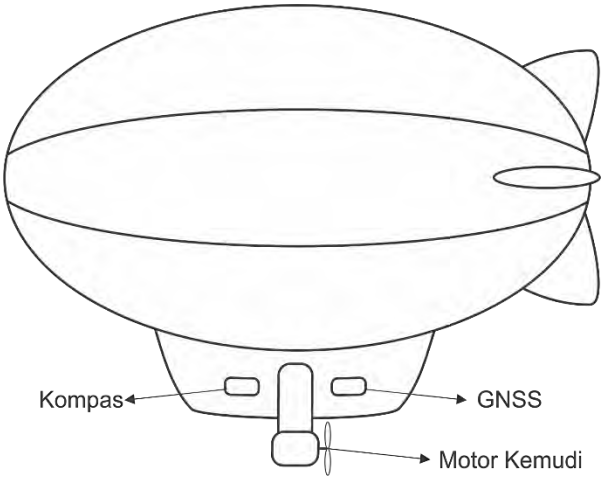
3.2.7. Motor DC

Penggerak balon udara berupa motor DC dengan tegangan 6 Volt. Motor DC yang digunakan adalah Syma X5C. Motor ini merupakan jenis motor bersikat. Arah pergerakan motor ditentukan oleh pemberian tegangan DC pada kedua input motor. Gambar 3.8 memberikan gambaran motor DC yang digunakan pada tugas akhir ini.

Gambar 3.8 Motor DC Syma X5C

3.2.8. Perancangan Posisi Motor dan Sensor

Peletakan motor dan sensor pada balon udara berada pada gondola yang dipasang pada titik tengah gravitasi balon. Motor kemudi dipasang di bagian tengah balon untuk memudahkan pergerakan balon saat berbelok. Sensor Kompas diletakkan pada bagian tengah depan balon udara untuk mendeteksi sudut arah gerak balon udara. Sensor GNSS (*Global Navigation Satellite System*) diletakkan pada bagian tengah belakang balon udara untuk mendeteksi posisi balon udara relatif terhadap bumi.



Gambar 3.9 Tata Letak Motor dan Sensor

3.3. Perancangan Perangkat Lunak

Pada tahapan ini, algoritma pemrograman dirancang untuk melakukan beberapa proses yang terbagi menjadi tiga bagian utama. Bagian pertama adalah akuisisi data sensor yaitu Kompas dan GNSS. Bagian kedua adalah perancangan sistem navigasi. Bagian terakhir adalah perancangan kontrol logika *fuzzy*.

3.1.1. Kalibrasi Magnetometer

Ketika mengakuisisi data magnetometer untuk dikonversi menjadi *heading*, perlu diketahui jenis kesalahan yang mempengaruhi pembacaan sensor. Kalibrasi diperlukan untuk melihat *offset* dari sensor dan meningkatkan akurasi nilai output sensor agar mendekati nilai sebenarnya.

Heading dapat ditentukan melalui penggunaan giroskop, akselerometer, dan magnetometer tiga sumbu yang tersedia di sensor IMU. Namun sistem kompas harus diterapkan dengan benar dan dapat mngkompensasi pengaruh elevasi dan kemiringan sudut.

Tanpa kalibrasi, data magnetometer akan terdistorsi sehingga pembacaan menjadi tidak tepat. Untuk melakukan kalibrasi dilakukan penghitungan *offset* yaitu dengan persamaan :

$$\alpha = \frac{x_{max}+x_{min}}{2} \tag{3.1}$$

$$\beta = \frac{y_{max}+y_{min}}{2} \tag{3.2}$$

$$\gamma = \frac{z_{max}+z_{min}}{2} \tag{3.3}$$

Hasil dari persamaan di atas adalah besar *offset* pada sumbu x, y, dan z. Kemudian pembacaan setiap sumbu magnetometer dikurangkan dengan hasil tersebut.

$$Mx = Mx - \alpha \tag{3.4}$$

$$My = My - \beta \tag{3.5}$$

$$Mz = Mz - \gamma \tag{3.6}$$

3.1.2. Kompensasi Kemiringan Kompas

Kompensasi kemiringan kompass sangat diperlukan pada navigasi udara. Jika sensor magnetometer dimiringkan, maka hasil pembacaan akan berubah karena terpengaruh oleh medan magnet bumi yang berubah

teradap gaya gravitasi bumi. Maka, diperlukan tambahan sensor yaitu akselerometer dan giroskop. Akselerometer akan mengukur percepatan gravitasi bumi sehingga derajat kemiringan dari sensor dapat diketahui. Sedangkan giroskop akan mengukur perubahan kecepatan sudut (dalam hal ini *yaw*) untuk mengkopensasi pembacaan *heading* kompas.

Dengan kompensasi kemiringan, maka arah dari kompas akan tetap pada arah sebenarnya dan tidak mengalami *offset*. Algoritma yang digunakan untuk mengkompensasi data kompas adalah AHRS (*Attitude Heading Reference System*). AHRS akan mengkonversi data keluaran magnetometer, akselerometer, dan giroskop yang telah dikalibrasi menjadi representasi kuarternion. Hasil dari konversi tersebut kemudian diproses pada filter kuarternion Magwick (Gambar 2.10). Filter tersebut menghasilkan empat elemen kuarternion yaitu q_0, q_1, q_2, dan q_3. Elemen kuarternion tersebut kemudian dikonversi menjadi sudut Euler menggunakan persamaan 2.6 yang menghasilkan keluaran berupa *roll* (Φ), *pitch* (θ), dan *yaw* (ψ). Sehingga didapatkan nilai *heading* dari hasil konversi kuarternion yang berupa *yaw*.

3.1.3. Akuisisi Data GNSS

GNSS *Receiver* memiliki output standar yang berisi informasi yang berhubungan dengan data-data geografi. Standar format informasi tersebut diberi nama NMEA (*National Marine Electronics Assosiation*) 0183. Beberapa ketentuan umum standar NMEA tersebut adalah :

1. Informasi NMEA dikirimkan oleh vendor dalam bentuk data teks dengan panjang maksimal 80 karakter.
2. Data teks NMEA berformat :

   "$<*vendor*><*massage*><*parameters*><*checksum*><CR><LF>"

3. Kombinasi <*vendor*><*massage*> disebut *address field*.
4. Kode *vendor* untuk GPS adalah "GP".

Standar NMEA 0183 memiliki banyak jenis bentuk kalimat laporan yang masing-masing mengandung data yang berbeda beda, di antaranya :

Tabel 3.1 Kode NMEA

Adress field	Informasi
$GPGGA	*Global positioning system fixed data*
$GPGLL	*Geographic position - latitude / longitude*
$GPGSA	*GNSS DOP and active satellites*
$GPGSV	*GNSS satellites in view*
$GPRMC	*Recommended minimum specific GNSS data*

Data yang diperlukan adalah data posisi yang berupa *lattitude* dan *longitude* sehingga kode informasi NMEA yang diperlukan adalah $GPGGA. $GPGGA adalah data posisi valid yang diterima sensor GNSS. $GPGGA memberikan informasi antara lain koordinat lintang, bujur, dan waktu. Contoh pesan $GPGGA adalah :

$GPGGA,132453.970,2651.0138,N,07547.7054,E,1

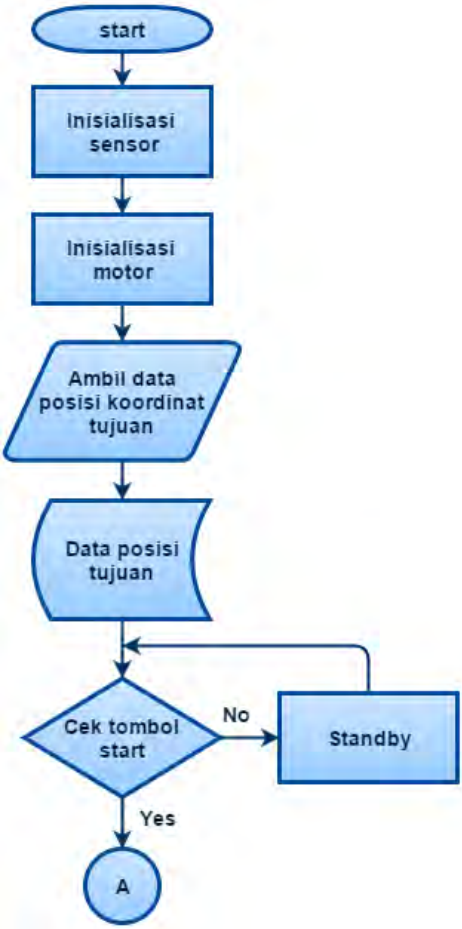
Dari data di atas dilakukan *parsing* data untuk mendapatkan informasi posisi. Klasifikasi yang diperlukan adalah :

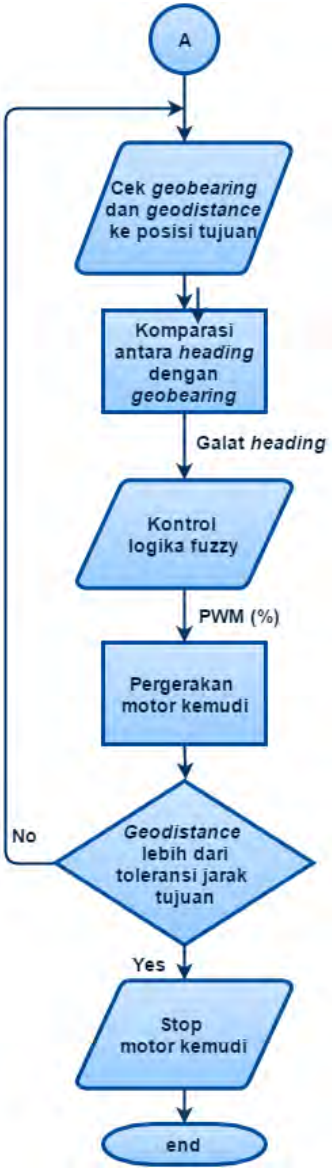
GGA	→	*Global Positioning System Fix Data*
132453.970	→	Data diambil saat 13:24:53970 UTC
2651.0138, N	→	*Latitude* 26 deg 51.0138' N
07547.7054, E	→	*Longitude* 07 deg 54.7054' E

3.1.4. Sistem Navigasi

Langkah awal yang dilakukan oleh sistem adalah inisialisasi sensor. Sensor yang digunakan adalah sensor GNSS dan sensor kompas. Kemudian setelah itu dilakukan inisialisasi motor. Ada 2 motor yang digunakan sebagai motor kemudi yang berada di samping balon. Setelah inisialisasi sensor dam motor, dilakukan pengambilan data *waypoint* yang dilakukan oleh user pada antarmuka. Data *waypoint* disimpan pada memori, kemudian balon udara tetap *standby* sampai tombol *start* ditekan. *Flow* data dan *flowchart* proses pada sistem ini ditunjukkan pada gambar 3.10 dan 3.11.

Gambar 3.10 Blok diagram *flow* data


start
Inisialisasi sensor
Inisialisasi motor
Ambil data posisi koordinat tujuan
Data posisi tujuan
Cek tombol start
No
Standby
Yes
A



Gambar 3.11 *Flowchart* sistem navigasi

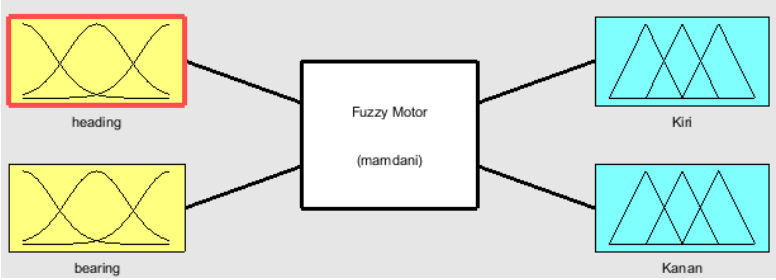
Ketika tombol *start* ditekan, maka balon udara akan melakukan perhitungan *geodistance* (jarak posisi sekarang dan posisi target) dan *geobearing* (sudut antara posisi sekarang dan posisi target relatif tehadap arah utara). *Geobearing* akan dibandingkan dengan *heading* (sudut arah hadap balon udara relatif tehadap arah utara) yang terdeteksi dari sensor kompas. Balon udara akan mengetahui berapa galat sudut *heading* terhadap *geobearing*. Nilai galat tersebut dimasukkan pada kontrol logika *fuzzy*. Kemudian balon udara akan mengaktifkan motor kemudi yang kecepatannya diatur dari keluaran kontrol logika *fuzzy* untuk bergerak ke arah tujuan berdasarkan galat sudut *heading* terhadap *geobearing*. Pergerakan tadi akan mempertahankan nilai galat *heading* terhadap *geobearing* bernilai nol sehingga *track* yang akan dilalui oleh balon terarah menuju ke *waypoint*. Hal tersebut akan dilakukan berulang selama *geodistance* kurang dari niai toleransi jarak tujuan yang ditentukan oleh user.

Jika nilai *geodistance* lebih dari nilai toleransi jarak tujuan, balon udara akan berhenti karena posisi yang terdeteksi oleh balon udara dianggap sudah mencapai area *waypoint*. Proses navigasi oleh balon udara dinyatakan selesai.

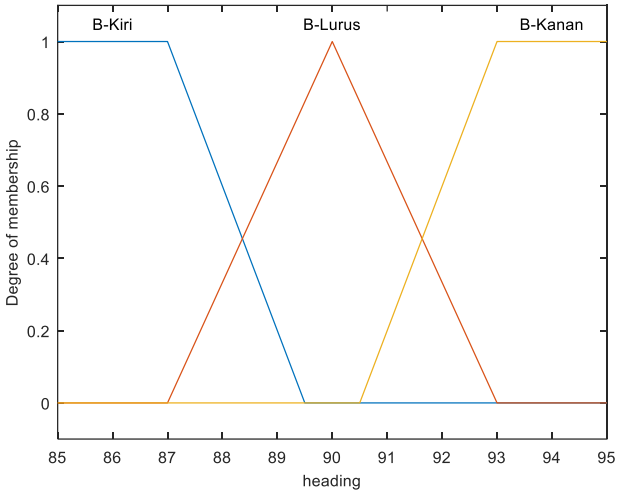
3.1.5. Kontrol *Fuzzy*

Perhitungan kecepatan motor dilakukan dengan menggunakan bantuan logika *fuzzy* untuk mempermudah proses penentuan. Pada sistem yang dirancang digunakan sebuah *Fuzzy Interference System* (FIS), yaitu FIS untuk kontrol motor samping.

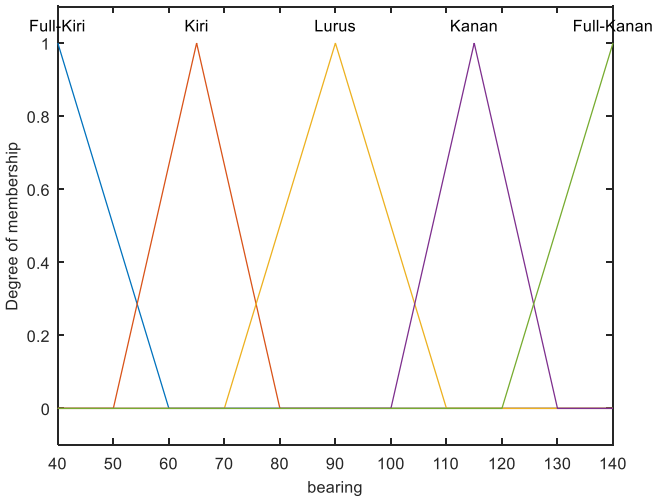
Untuk FIS motor samping dirancang *fuzzy* 2 input 2 output, untuk lebih jelasnya dapat dilihat di gambar 3.12 – gambar 3.16.



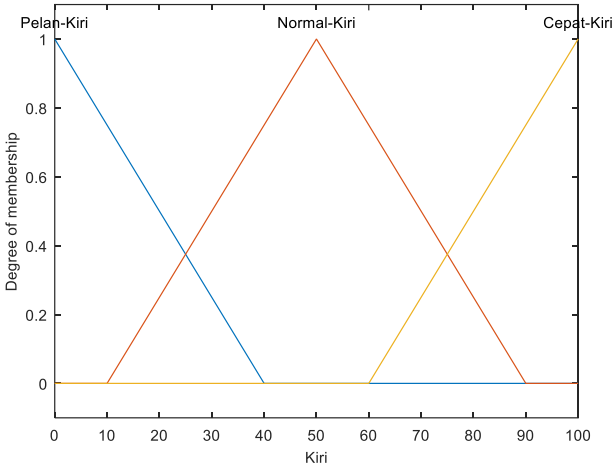
Gambar 3.12 Sistem *fuzzy* yang dirancang untuk motor samping



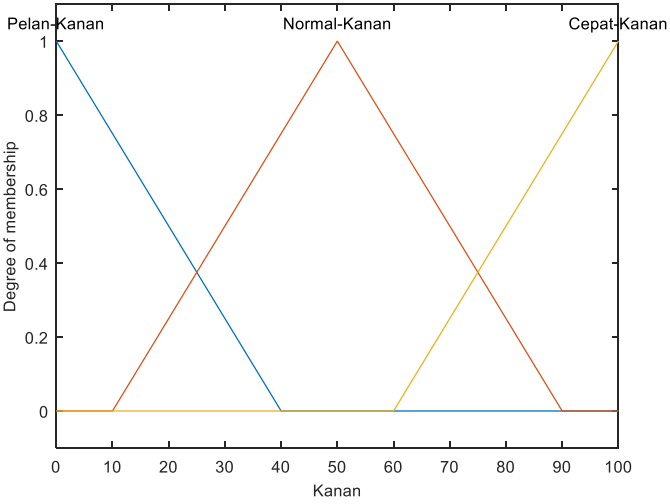
Gambar 3.13 Grafik fungsi keanggotaan untuk *heading*



Gambar 3.14 Grafik fungsi keanggotaan untuk *bearing*



Gambar 3.15 Grafik fungsi keanggotaan untuk motor kiri



Gambar 3.16 Grafik fungsi keanggotaan untuk motor kanan

Tabel 3.2 *Fuzzy Rule* untuk Motor Samping

e-posisi / arah	Fkiri	Kiri	Tengah	Kanan	Fkanan
eKiri	PKi, CKa	NKi, NKa	NKi, CKa	PKi, CKa	CKi, PKa
eLurus	PKi, CKa	NKi, CKa	CKa, CKa	NKi, CKa	CKi, PKa
eKanan	PKi, CKa	CKi, PKa	CKa, CKi	NKi, NKa	CKi, PKa

Keterangan:
CKi = Cepat Kiri,
NKi = Normal Kiri,
PKi = Pelan Kiri,
CKi = Cepat Kiri,
NKi = Normal Kiri,
PKa = Pelan Kanan.

Perhitungan *fuzzy* menggunakan metode mamdani dengan menggunakan operasi MIN-MAX dan untuk *defuzzyfication*-nya menggunakan operasi *centroid*. Semua output berupa pwm kecepatan motor. Dan semua operasi *fuzzy* dilakukan pada Arduino Mega.

BAB IV
PENGUJIAN DAN ANALISIS

Pada bab ini akan dibahas mengenai pengujian dari sistem yang telah dirancang. Bab ini bertujuan untuk mengetahui apakah tujuan dalam perancangan sistem pada tugas akhir ini telah terlaksana atau tidak. Pengujian pada bab ini terdiri dari pengujian berat balon, pengujian keseimbangan balon, kalibrasi kompas, kompensasi kemiringan kompas dengan AHRS, GNSS, dan pengujian sistem navigasi mempertahankan arah.

4.1. Pengujian Berat Beban Maksimal Balon

Pada tugas akhir ini digunakan dua balon udara *air swimmer* yang diisi dengan helium dengan ukuran panjang 110 cm, lebar 60 cm, dan tinggi 60 cm.. Volume dari balon udara ini adalah 0,1274 m^3. Dua balon udara *air swimmer* mempunyai volume untuk diisi helium sebesar 0,2548 m^3.

Helium utamanya digunakan sebagai gas pengangkat pada balon udara. Jenis gas lain yang dapat digunakan sebagai gas pengangkat pada balon udara adalah gas hidrogen. Volume 1 m^3 gas hidrogen dapat mengangkat beban hingga 1088 gram. Namun gas hidrogen sangat mudah terbakar, sehingga digunakan gas helium.

Gambar 4.1 Balon udara yang digunakan

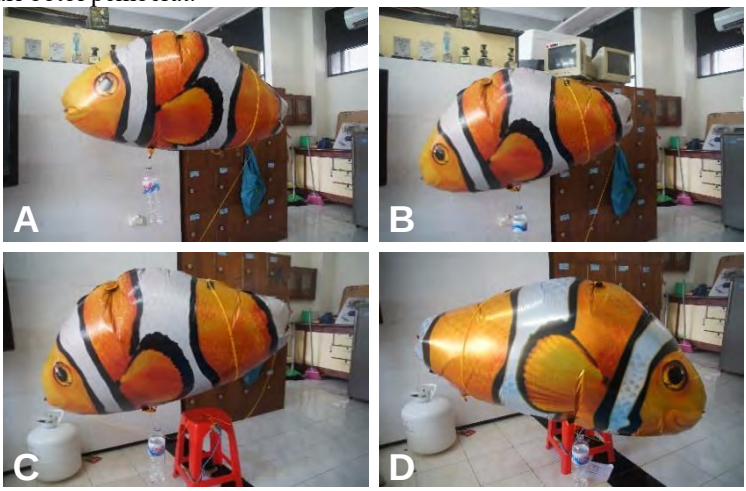
Tabel 4.1 Daya Angkat Gas Hidrogen dan Helium

Gas	Volume Gas (m^3)	Beban Maksimum (gram)
Hidrogen	0,2548	277,2
Helium	0,2548	244,6

Perbandingan gaya angkat gas helium dengan gas hidrogen seperti terlihat pada tabel 4.1. Dengan isian 1 m^3 helium dapat mengangkat beban 960 gram. Sehingga balon udara *air swimmer* dengan volume 0.2548 m^3 dapat mengangkat beban sekitar 240 gram jika diisi penuh gas helium.

Pengujian berat beban maksimal balon dilakukan dengan cara memasang pemberat pada balon hingga balon tidak dapat terbang di udara. Pemberat yang digunakan adalah berupa air yang ditempatkan pada botol pemberat. Air dalam botol pemberat akan diisi perlahan hingga botol pemberat menyentuh lantai. Kemudian dilakukan pengukuran berat dari air yang ada dalam botol pemberat.

Pada gambar 4.2 (A) botol pemberat masih belum terisi air dan balon masih terbang. Berat dari botol dalam keadaan kosong adalah 33 gram (Gambar 4.3). Kemudian botol pemberat diisi dengan air pada gambar 4.2 (B) dan (C). Terlihat balon mulai melayang dan turun. Pada gambar 4.2 (D) botol pemberat sudah menyentuh lantai. Dilakukan pengukuran berat dari botol pemberat.



Gambar 4.2 Pengujian beban maksimal balon

Gambar 4.3 Berat kosong dari botol pemberat

Gambar 4.4 Pengkuran berat maksimal

Terlihat pada gambar 4.4 bahwa berat dari botol pemberat yang telah terisi air adalah 233 gram. Berat tersebut berbeda sebesar 7 gram dari gaya angkat gas helium sesuai perbandingan pada tabel 4.1 yaitu 240 gram.

4.2. Pengujian Keseimbangan Balon

Gambar 4.5 Beban pada jarak 10 cm dari depan balon

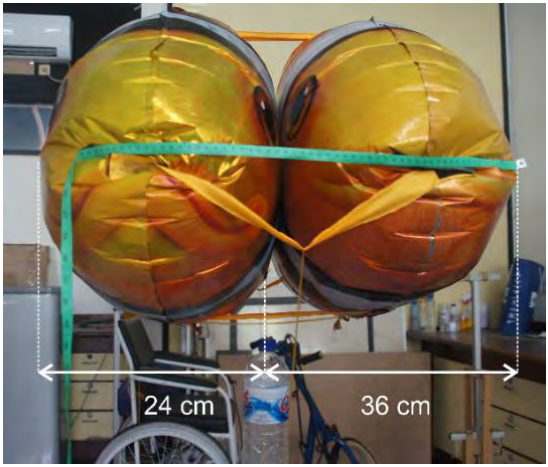
Pada pengujian keseimbangan, diletakkan beban pada balon untuk mengetahui titik tengah gravitasi balon. Pengujian pertama adalah pengujian keseimbangan balon terhadap sudut *pitch*. Pada gambar 4.5, diletakkan beban pada jarak 10 cm dari depan balon. Terlihat badan balon condong ke depan. Sehingga diuji lagi dengan menggeser beban ke arah tengah balon.

Gambar 4.6 Beban pada jarak 20 cm dari depan balon

Gambar 4.7 Beban pada jarak 30 cm dari depan balon

Pada gambar 4.6, diletakkan beban pada jarak 20 cm dari depan balon. Terlihat balon masih condong ke depan. Maka dilakukan lagi penggeseran beban.

Pada gambar 4.7 diletakkan beban pada jarak 30 cm dari depan balon. Terlihat balon sudah tidak miring ke depan. Pengujian kedua adalah pengujian keseimbangan balon terhadap sudut *roll*. Pengujian dilakukan dengan menggeser beban pada pengujian sudut *pitch* terakhir ke antara 2 balon.



Gambar 4.8 Keseimbangan terhadap sudut *roll*

Pada gambar 4.8 diletakkan beban di antara 2 balon. Terlihat bahwa balon sudah seimbang. Sehingga dapat disimpulkan bahwa titik tengah gravitasi balon pada pada jarak 30 cm dari depan balon, 24 cm dari kanan balon, dan 36 cm dari kiri balon. Pada titik ini dilakukan pemasangan kotak sensor.

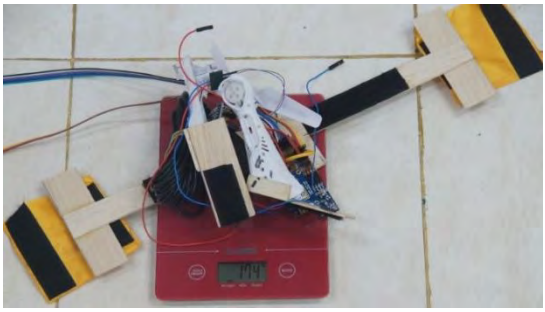
4.3. Pengujian Berat Komponen

Pengujian berat komponen diperlukan karena ada batasan berat masksimal yang dapat diangkat oleh balon udara. Pengujian berat dilakukan pada masing-masing komponen yang akan dipasang pada balon udara yaitu 2 buah motor kemudi, kotak sensor, sensor kompas, sensor GNSS, Arduino Pro Mini, kabel, papan rangkaian, dan antena GNSS. Berikut ini hasil pengukuran tersebut :

Jumlah total berat komponen pada tabel 4.2 adalah 172 gram. Pengukuran total seperti terlihat pada gambar 4.9. Berat yang terukur pada timbangan adalah 174 gram.

Tabel 4.2 Berat Komponen

Komponen	Berat (gram)
2 motor kemudi	28
Kotak sensor	9
Sensor kompas	2
Sensor GNSS	8
Arduino Pro Mini	4
Kabel	8
Papan rangkaian	9
Antena GNSS	104

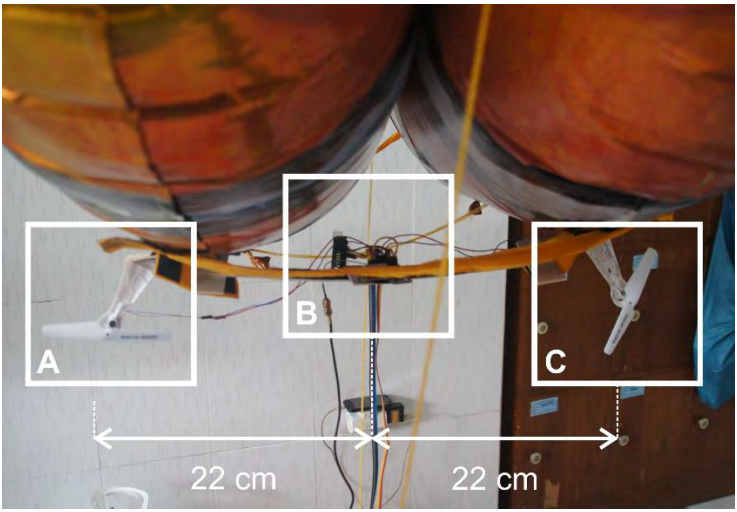


Gambar 4.9 Total berat komponen

4.4. Reaslisasi Desain Balon Udara

Gambar 4.10 Realisasi sistem navigasi pada balon (tampak samping)

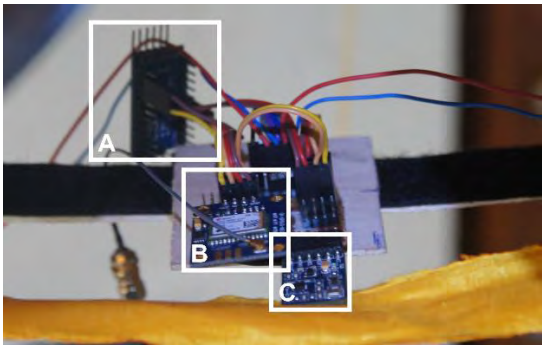
Gambar 4.11 Realisasi sistem navigasi pada balon (tampak depan)



Gambar 4.12 Realisasi kotak sensor dan lokasi motor

Keterangan:
A = Motor Samping Kanan
B = Kotak Sensor
C = Motor Samping Kiri

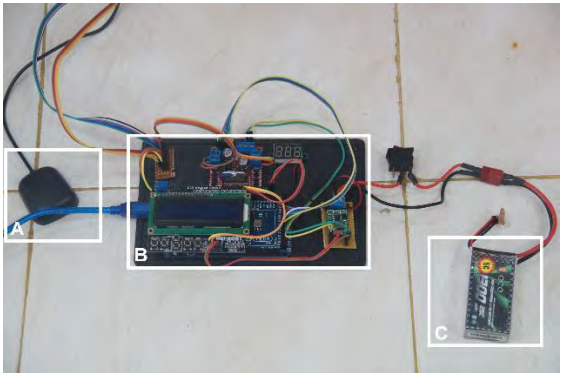
Setelah dilakukan pemasangan komponen, balon udara masih melayang di udara. Hal tersebut menunjukkan gaya beban lebih kecil dari gaya angkat gas helium pada balon udara. Gambar 4.12 menunjukkan lokasi daripada sensor beserta motor yang digunakan. Pada alat yang dibuat, digunakan satu set motor samping yang berguna untuk motor kemudi. Hal ini digunakan untuk mempermudah balon udara dalam berbelok. Posisi dari kotak sensor dan motor ke tepat pada titik tengah gravitasi pada balon udara sehingga balon udara dapat tetap seimbang. Posisi kotak sensor juga sejajar dengan pemasangan motor kemudi agar sudut arah hadap balon yang terdeteksi oleh sensor kompas sesuai dengan respon mmotor kemudi saat balon bergerak. Selain sensor kompas, pada kotak sensor juga terdapat sensor GNSS untuk mendeteksi posisi balon dan Arduino Pro Mini untuk mengakuisisi data sensor GNSS. Posisi sensor pada balon udara seperti terlihat pada gambar 4.13.



Gambar 4.13 Posisi sensor

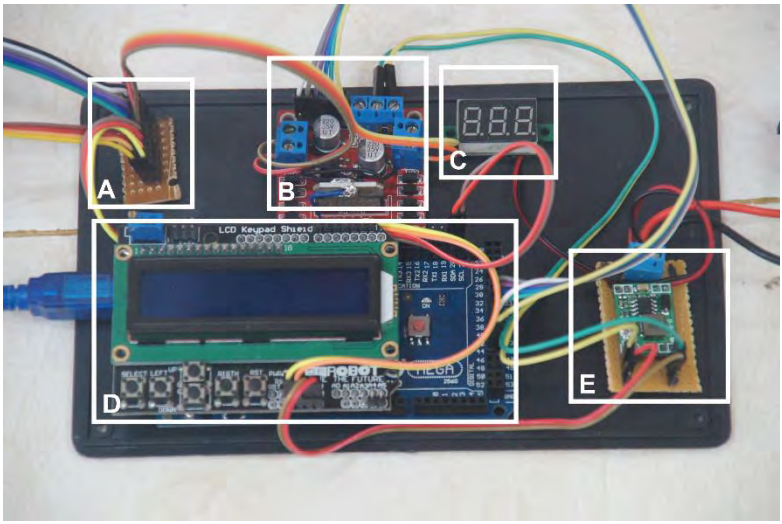
Keterangan:
A = Arduino Pro Mini
B = Sensor GNSS
C = Sensor Kompas

Karena keterbatasan beban yang dapat diangkat oleh balon udara, maka posisi antena GNSS, kontoller dan suplai daya diletakkan terpisah (Gambar 4.14). Kontroller tersambung ke kotak sensor menggunakan kabel.



Gambar 4.14 Kontroller dan suplai daya

Keterangan:
A = Antena GNSS
B = Kontroller
C = Baterai Li-Po



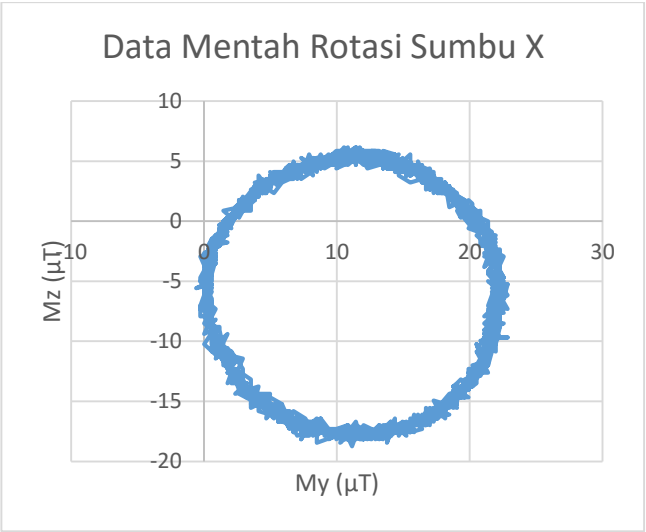
Gambar 4.15 Komponen kontroller

Keterangan:
A = *Jumper*
B = Driver Motor
C = Indikator Baterai
D = Arduino Mega
E = *Buck Converter*

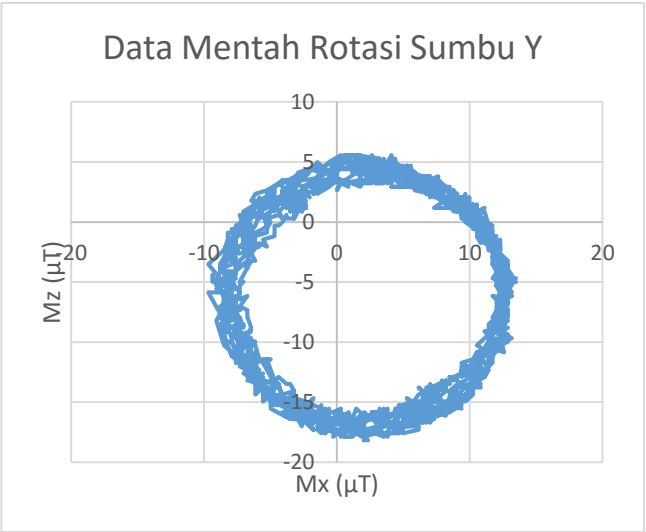
Gambar 4.15 menunjukkan beberapa komponen kontroller. *Buck Converter* dengan tegangan output sebesar 7 V digunakan sebagai suplai daya driver motor. Koneksi Arduino Mega dengan sensor kompas dan GNSS dilakukan dengan antarmuka I2C.

4.5. Kalibrasi Sensor Magnetometer

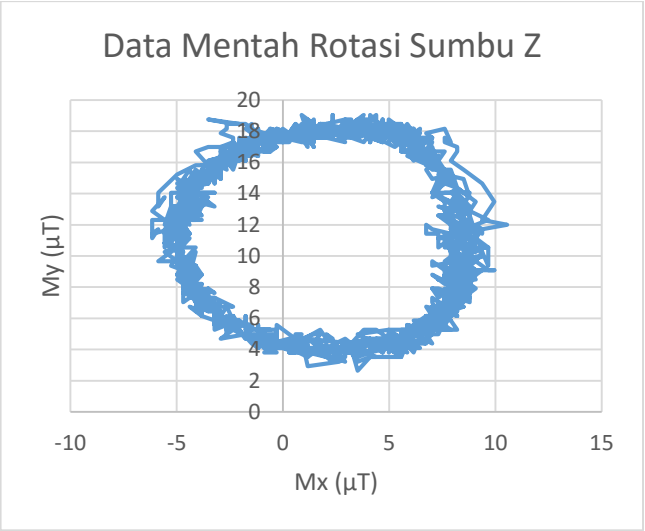
Kalibrasi sensor magnetometer sangat diperlukan untuk mengetahui besar galat pembacaan medan magnet bumi. Kalibrasi juga digunakan untuk meningkatkan akurasi pembacaan sensor. Kalibrasi dilakukan dengan memutar sensor pada setiap sumbu untuk mendapatkan data mentah sensor.



Gambar 4.16 Data mentah sensor pada rotasi sumbu X



Gambar 4.17 Data mentah sensor pada rotasi sumbu Y



Gambar 4.18 Data mentah sensor pada rotasi sumbu Z

Gambar 4.16 - 4.18 di atas adalah penjabaran data sensor yang diputar pada sumbu x, y, dan z. Terlihat posisi lingkaran tidak tepat di titik 0,0. Hal tersebut dapat mengakibatkan kesalahan dalam pembacaan derajat arah dari sensor. Untuk menentukan besar *offset* yang diperukan agar posisi lingkaran menjadi di titik 0,0 adalah menggunakan persamaan 3.1 - 3.3.

Program untuk melakukan proses tersebut adalah :

```
1.  sample_count = 128;
2.  for(ii = 0; ii < sample_count; ii++)
3.  {
4.   readMagData(mag_temp);
5.   for (int jj = 0; jj < 3; jj++)
6.   {
7.     if(mag_temp[jj] > mag_max[jj]) mag_max[jj]
           = mag_temp[jj];
8.     if(mag_temp[jj] < mag_min[jj]) mag_min[jj]
           = mag_temp[jj];
9.   }
10. }
```

Hasil dari proses di atas adalah besar *offset* pada sumbu x, y, dan z. Kemudian pembacaan setiap sumbu kompas dikurangkan dengan hasil tersebut sesuai dengan persamaan 3.4 - 3.6.

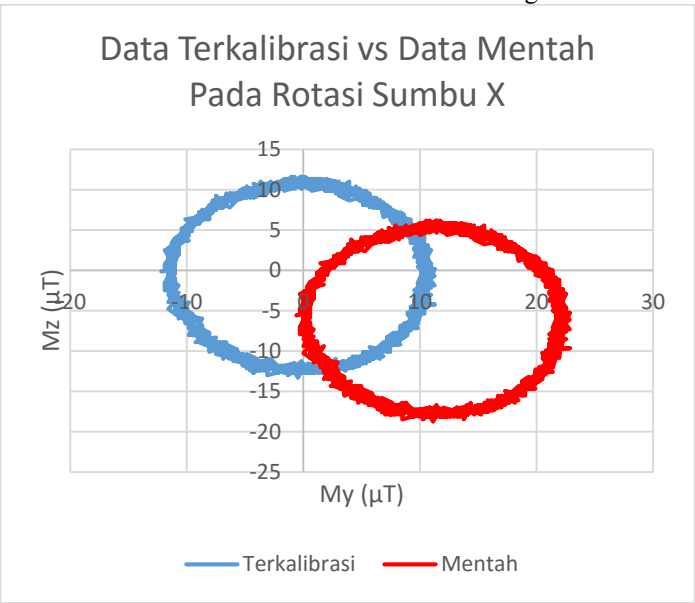
Program untuk melakukan proses tersebut adalah :

```
1.  mx = (float)magCount[0]*mRes*magCalibration[0]
          - compCalib.magBias[0];
2.  my = (float)magCount[1]*mRes*magCalibration[1]
          - compCalib.magBias[1];
3.  mz = (float)magCount[2]*mRes*magCalibration[2]
          - compCalib.magBias[2];
```

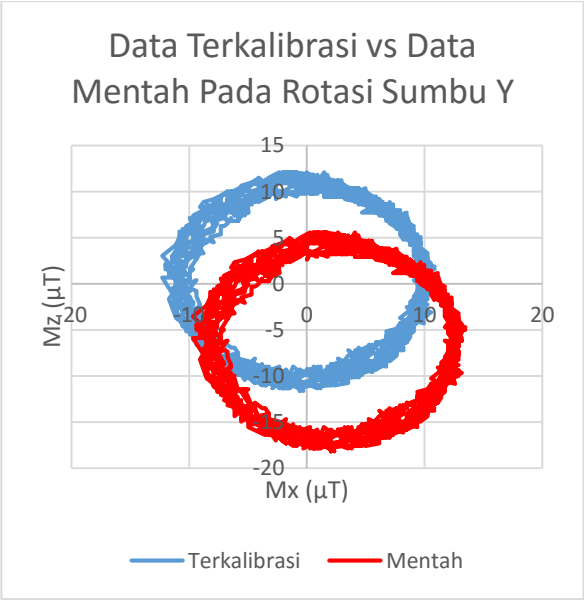
Didapatkan data bahwa besar *offset* adalah :

x = 2,354 mT
y = 12,795 mT
z = -9,283 mT

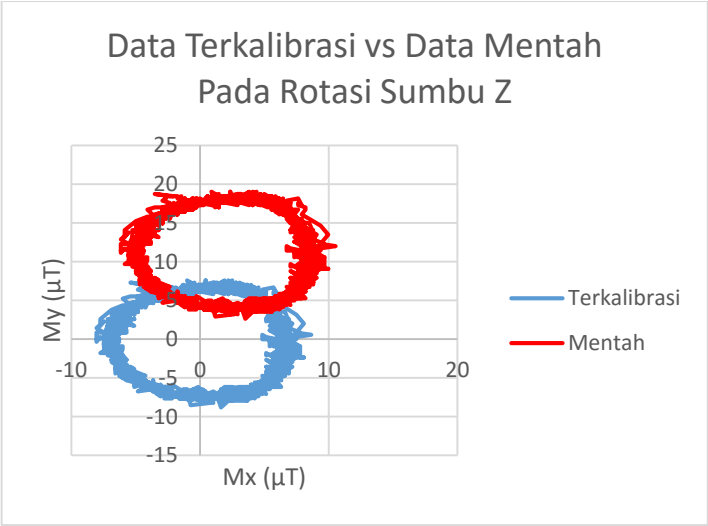
Berikut ini adalah hasil kalibrasi dari sensor magetometer :



Gambar 4.19 Hasil kalibrasi pada sumbu X



Gambar 4.20 Hasil kalibrasi pada sumbu Y



Gambar 4.21 Hasil kalibrasi pada sumbu Z

Terlihat dari sebaran data sensor ketika diputar dalam sumbu x, y, dan z setelah dikalibrasi, posisi pusat lingkaran menjadi di titik 0,0. Dengan demikian jika data hasil kalibrasi tersebut dikonversi menjadi derajat arah akan menjadi menjadi lebih akurat.

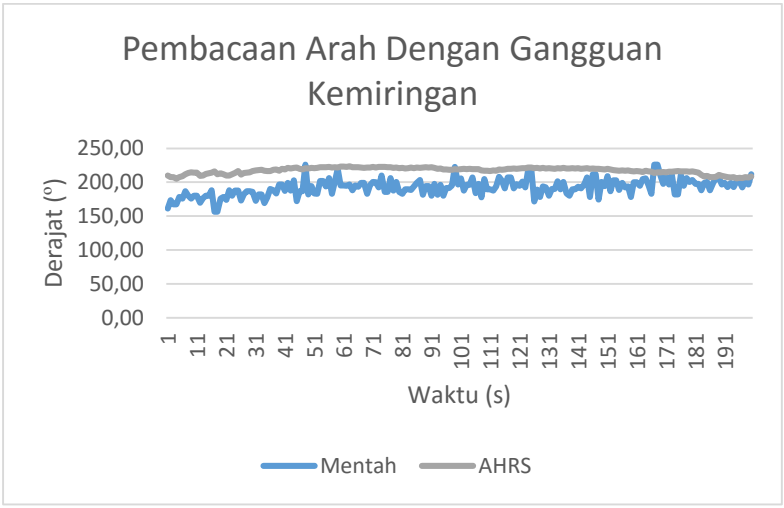
4.6. Kompensasi Kemiringan Kompas

Pengujian kompensasi kemiringan kompas dilakukan dengan menggerakkan sensor pada kemiringan tertentu pada penunjukkan arah yang sama (Gambar 4.22). Sehingga akan terlihat perbedaan antara arah sebenarnya dengan pembacaan sensor.

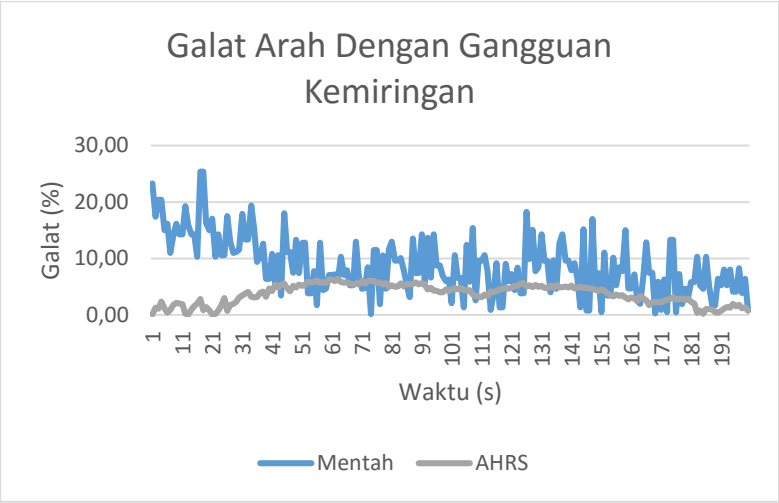
Algoritma kompensasi kemiringan kompas yang digunakan pada tugas akhir ini adalah algoritma kuarternion Magwick yang berbasis AHRS (*Attitude And Heading Reference System*). Algoritma kompensasi kemiringan kompas dilakukan pada proses di bawah ini :

```
1.  MadgwickQuaternionUpdate(-ay,-ax,az,gy*PI/180.0f,
    gx*PI/180.0f,-gz*PI/180.0f,mx,my,mz);
2.  yaw   = atan2(2.0f*(q[1]*q[2]+q[0]*q[3]),q[0]*q[0]
            +q[1]*q[1]-q[2]*q[2]-q[3]*q[3]);
3.  pitch = -asin(2.0f*(q[1]*q[3]-q[0]*q[2]));
4.  roll  = atan2(2.0f*(q[0]*q[1]+q[2]*q[3]),q[0]*q[0]
            -q[1]*q[1]-q[2]*q[2]+q[3]*q[3]);
5.
6.  pitch *= 180.0f / PI;
7.  yaw   *= 180.0f / PI;
8.  yaw   -= 1.3f; // Declination at Surabaya
9.  roll  *= 180.0f / PI;
10.
11. heading = yaw;
12. heading -= 180;
13.
14. while (heading < -180) heading += 360;
15. while (heading >  180) heading -= 360;
```

Gambar 4.22 Pengujian kompensasi kemiringan



Gambar 4.23 Output dari algoritma AHRS



Gambar 4.24 Galat algoritma AHRS

Terlihat bahwa galat dari pembacaan arah sensor relatif kecil setelah algoritma mencapai *steady state*. Dari 200 data yang didapatkan, maksimum galat yang terukur adalah 6,41%. Terbukti bahwa filter kuarternion AHRS dapat meminimalisasi galat pembacaan arah dari sensor.

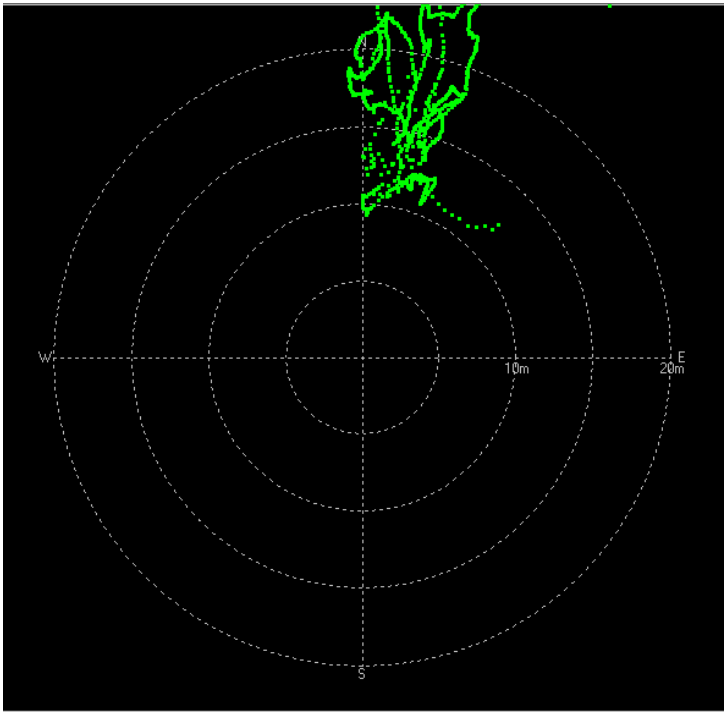
4.7. Pengujian GNSS

Pengujian sensor GNSS dilakukan dengan cara mengambil data koordinat selama 10 menit intuk mengetahui rentang deviasi koordinat yang terdeteksi sensor dengan posisi sebenarnya.

Proses pengambilan data koordinat dilakukan pada proses di bawah ini :

```
1.  void parseDegrees(const char *term, RawDegrees °)
2.  {
3.    uint32_t leftOfDecimal = (uint32_t)atol(term);
4.    uint16_t minutes = (uint16_t)(leftOfDecimal%100);
5.    uint32_t multiplier = 10000000UL;
6.    uint32_t tenMillionthsOfMinutes=minutes*
       multiplier;
7.
8.    deg.deg = (int16_t)(leftOfDecimal / 100);
```

```
9.
10.   while (isdigit(*term))
11.     ++term;
12.
13.   if (*term == '.')
14.     while (isdigit(*++term))
15.     {
16.       multiplier /= 10;
17.       tenMillionthsOfMinutes+=(*term-'0')*
           multiplier;
18.     }
19.
20.   deg.billionths=(5*tenMillionthsOfMinutes+1) / 3;
21.   deg.negative = false;
22. }
```



Gambar 4.25 Deviasi Sensor GNSS

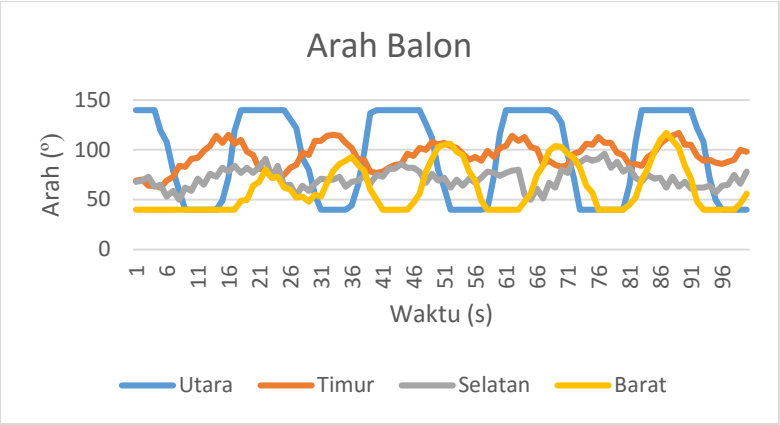
Gambar 4.25 menunjukkan hasil pengujian pengukuran posisi sensor GNSS. Hasil yang didapatkan menunjukkan posisi yang terdeteksi rata-rata berada pada rentang 9,5-20 meter dari posisi sebenarnya. Hal ini disebabkan karena sensor GNSS yang digunakan memiliki spesifikasi akurasi 10-25 meter.

4.8. Pengujian Sistem Navigasi Mempertahankan Arah

Pengujian ini dilakukan untuk mengetahui respon sistem navigasi untuk mempertahankan arah yang diinputkan oleh user.

Sistem navigasi dilakukan pada proses di bawah ini :

```
1.  if ( tagLoc.lat == 0 || currentLoc.lat == 0 ) {
2.      berhentiLR();
3.      return;
4.    }
5.
6.    int autoSteer = 90;
7.
8.    float turn = bearing - heading;
9.    while (turn < -180) turn += 360;
10.   while (turn >  180) turn -= 360;
11.
12.   autoSteer = map(2 * turn, 180, -180, 0, 180);
13.   autoSteer = constrain(autoSteer, 40, 140);
14.
15.   float angleError = abs(turn);
16.
17.   Serial.println(autoSteer);
18.   fuzzy_loop(autoSteer);
19.
20.   if (distance <= Waypoint_Dist_Tolerance)
21.   {
22.     berhentiLR();
23.     Serial.print("Arrived");
24.   }
```



Gambar 4.26 Grafik Arah Balon

Pada gambar di atas terlihat bahwa respon sistem masih berosilasi untuk mempertahankan arah Utara dan Barat. Simpangan osilasi arah Utara berkisar antara 50-90 derajat. Sedangkan simpangan osilasi arah barat sebesar 20-50 derajat. Namun untuk arah Timur dan Selatan, sistem tidak menghasilkan galat yang terlalu besar.

LAMPIRAN

Kode algoritma kuarternion Magwick :

```
1.  void MadgwickQuaternionUpdate(float ax, float ay, f
    loat az, float gx, float gy, float gz, float mx, fl
    oat my, float mz)
2.  {
3.      float q1 = q[0], q2 = q[1], q3 = q[2], q4 = q[3
    ];
4.      float norm;
5.      float hx, hy, _2bx, _2bz;
6.      float s1, s2, s3, s4;
7.      float qDot1, qDot2, qDot3, qDot4;
8.
9.      float _2q1mx;
10.     float _2q1my;
11.     float _2q1mz;
12.     float _2q2mx;
13.     float _4bx;
14.     float _4bz;
15.     float _2q1 = 2.0f * q1;
16.     float _2q2 = 2.0f * q2;
17.     float _2q3 = 2.0f * q3;
18.     float _2q4 = 2.0f * q4;
19.     float _2q1q3 = 2.0f * q1 * q3;
20.     float _2q3q4 = 2.0f * q3 * q4;
21.     float q1q1 = q1 * q1;
22.     float q1q2 = q1 * q2;
23.     float q1q3 = q1 * q3;
24.     float q1q4 = q1 * q4;
25.     float q2q2 = q2 * q2;
26.     float q2q3 = q2 * q3;
27.     float q2q4 = q2 * q4;
28.     float q3q3 = q3 * q3;
29.     float q3q4 = q3 * q4;
30.     float q4q4 = q4 * q4;
31.
32.     norm = sqrt(ax * ax + ay * ay + az * az);
33.     if (norm == 0.0f) return; // handle NaN
34.     norm = 1.0f/norm;
35.     ax *= norm;
```

```
36.     ay *= norm;
37.     az *= norm;
38.
39.     norm = sqrt(mx * mx + my * my + mz * mz);
40.     if (norm == 0.0f) return; // handle NaN
41.     norm = 1.0f/norm;
42.     mx *= norm;
43.     my *= norm;
44.     mz *= norm;
45.
46.     _2q1mx = 2.0f * q1 * mx;
47.     _2q1my = 2.0f * q1 * my;
48.     _2q1mz = 2.0f * q1 * mz;
49.     _2q2mx = 2.0f * q2 * mx;
50.     hx = mx * q1q1 - _2q1my * q4 + _2q1mz * q3 + mx
              * q2q2 + _2q2 * my * q3 + _2q2 * mz * q4 -
              mx * q3q3 - mx * q4q4;
51.     hy = _2q1mx * q4 + my * q1q1 - _2q1mz * q2 + _2
             q2mx * q3 - my * q2q2 + my * q3q3 + _2q3 *
             mz * q4 - my * q4q4;
52.     _2bx = sqrt(hx * hx + hy * hy);
53.     _2bz = -
             _2q1mx * q3 + _2q1my * q2 + mz * q1q1 + _2q
             2mx * q4 - mz * q2q2 + _2q3 * my * q4 - mz
             * q3q3 + mz * q4q4;
54.     _4bx = 2.0f * _2bx;
55.     _4bz = 2.0f * _2bz;
56.
57.     s1 = -
             _2q3 * (2.0f * q2q4 - _2q1q3 - ax) + _2q2 *
              (2.0f * q1q2 + _2q3q4 - ay) - _2bz * q3 *
             (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4
              - q1q3) - mx) + (-
             _2bx * q4 + _2bz * q2) * (_2bx * (q2q3 - q1
             q4) + _2bz * (q1q2 + q3q4) - my) + _2bx * q
             3 * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f -
             q2q2 - q3q3) - mz);
58.     s2 = _2q4 * (2.0f * q2q4 - _2q1q3 - ax) + _2q1
             * (2.0f * q1q2 + _2q3q4 - ay) - 4.0f * q2 *
              (1.0f - 2.0f * q2q2 - 2.0f * q3q3 - az) +
             _2bz * q4 * (_2bx * (0.5f - q3q3 - q4q4) +
             _2bz * (q2q4 - q1q3) - mx) + (_2bx * q3 + _
             2bz * q1) * (_2bx * (q2q3 - q1q4) + _2bz *
```

```
            (q1q2 + q3q4) - my) + (_2bx * q4 - _4bz * q
            2) * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f -
             q2q2 - q3q3) - mz);
59.     s3 = -
            _2q1 * (2.0f * q2q4 - _2q1q3 - ax) + _2q4 *
             (2.0f * q1q2 + _2q3q4 - ay) - 4.0f * q3 *
            (1.0f - 2.0f * q2q2 - 2.0f * q3q3 - az) + (
            -
            _4bx * q3 - _2bz * q1) * (_2bx * (0.5f - q3
            q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx) + (
            _2bx * q2 + _2bz * q4) * (_2bx * (q2q3 - q1
            q4) + _2bz * (q1q2 + q3q4) - my) + (_2bx *
            q1 - _4bz * q3) * (_2bx * (q1q3 + q2q4) + _
            2bz * (0.5f - q2q2 - q3q3) - mz);
60.     s4 = _2q2 * (2.0f * q2q4 - _2q1q3 - ax) + _2q3
            * (2.0f * q1q2 + _2q3q4 - ay) + (-
            _4bx * q4 + _2bz * q2) * (_2bx * (0.5f - q3
            q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx) + (
            -
            _2bx * q1 + _2bz * q3) * (_2bx * (q2q3 - q1
            q4) + _2bz * (q1q2 + q3q4) - my) + _2bx * q
            2 * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f -
            q2q2 - q3q3) - mz);
61.     norm = sqrt(s1 * s1 + s2 * s2 + s3 * s3 + s4 *
    s4);
62.     norm = 1.0f/norm;
63.     s1 *= norm;
64.     s2 *= norm;
65.     s3 *= norm;
66.     s4 *= norm;
67.
68.     qDot1 = 0.5f * (-
    q2 * gx - q3 * gy - q4 * gz) - beta * s1;
69.     qDot2 = 0.5f * (q1 * gx + q3 * gz - q4 * gy) -
    beta * s2;
70.     qDot3 = 0.5f * (q1 * gy - q2 * gz + q4 * gx) -
    beta * s3;
71.     qDot4 = 0.5f * (q1 * gz + q2 * gy - q3 * gx) -
    beta * s4;
72.
73.     q1 += qDot1 * deltat;
74.     q2 += qDot2 * deltat;
75.     q3 += qDot3 * deltat;
```

```
76.     q4 += qDot4 * deltat;
77.     norm = sqrt(q1 * q1 + q2 * q2 + q3 * q3 + q4 *
    q4);
78.     norm = 1.0f/norm;
79.     q[0] = q1 * norm;
80.     q[1] = q2 * norm;
81.     q[2] = q3 * norm;
82.     q[3] = q4 * norm;
83. }
```

Kode algoritma kontrol logika fuzzy :

```
1.  #define FIS_TYPE float
2.  #define FIS_RESOLUSION 101
3.  #define FIS_MIN -3.4028235E+38
4.  #define FIS_MAX 3.4028235E+38
5.  typedef FIS_TYPE(*_FIS_MF)(FIS_TYPE, FIS_TYPE*);
6.  typedef FIS_TYPE(*_FIS_ARR_OP)(FIS_TYPE, FIS_TYPE);

7.  typedef FIS_TYPE(*_FIS_ARR)(FIS_TYPE*, int, _FIS_AR
    R_OP);
8.
9.  const int fis_gcI = 2;
10. const int fis_gcO = 2;
11. const int fis_gcR = 13;
12.
13. FIS_TYPE g_fisInput[fis_gcI];
14. FIS_TYPE g_fisOutput[fis_gcO];
15.
16. FIS_TYPE fis_trapmf(FIS_TYPE x, FIS_TYPE* p)
17. {
18.     FIS_TYPE a = p[0], b = p[1], c = p[2], d = p[3]
    ;
19.     FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0 : ((
    c != d) ? ((d - x) / (d - c)) : 0)));
20.     FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0 : ((
    a != b) ? ((x - a) / (b - a)) : 0)));
21.     return (FIS_TYPE) min(t1, t2);
22. }
23.
24. FIS_TYPE fis_trimf(FIS_TYPE x, FIS_TYPE* p)
25. {
```

```
26.     FIS_TYPE a = p[0], b = p[1], c = p[2];
27.     FIS_TYPE t1 = (x - a) / (b - a);
28.     FIS_TYPE t2 = (c - x) / (c - b);
29.     if ((a == b) && (b == c)) return (FIS_TYPE) (x
    == a);
30.     if (a == b) return (FIS_TYPE) (t2*(b <= x)*(x <
    = c));
31.     if (b == c) return (FIS_TYPE) (t1*(a <= x)*(x <
    = b));
32.     t1 = min(t1, t2);
33.     return (FIS_TYPE) max(t1, 0);
34. }
35.
36. FIS_TYPE fis_min(FIS_TYPE a, FIS_TYPE b)
37. {
38.     return min(a, b);
39. }
40.
41. FIS_TYPE fis_max(FIS_TYPE a, FIS_TYPE b)
42. {
43.     return max(a, b);
44. }
45.
46. FIS_TYPE fis_array_operation(FIS_TYPE *array, int s
    ize, _FIS_ARR_OP pfnOp)
47. {
48.     int i;
49.     FIS_TYPE ret = 0;
50.
51.     if (size == 0) return ret;
52.     if (size == 1) return array[0];
53.
54.     ret = array[0];
55.     for (i = 1; i < size; i++)
56.     {
57.         ret = (*pfnOp)(ret, array[i]);
58.     }
59.
60.     return ret;
61. }
62.
63. _FIS_MF fis_gMF[] =
64. {
```

```
65.     fis_trapmf, fis_trimf
66. };
67.
68. int fis_gIMFCount[] = { 3, 5 };
69.
70. int fis_gOMFCount[] = { 3, 3 };
71.
72. FIS_TYPE fis_gMFI0Coeff1[] = { 80.5, 84.5, 87, 89.5
     };
73. FIS_TYPE fis_gMFI0Coeff2[] = { 87, 90, 93 };
74. FIS_TYPE fis_gMFI0Coeff3[] = { 90.5, 93, 95.5, 99.5
     };
75. FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1, fis
    _gMFI0Coeff2, fis_gMFI0Coeff3 };
76. FIS_TYPE fis_gMFI1Coeff1[] = { 15, 40, 60 };
77. FIS_TYPE fis_gMFI1Coeff2[] = { 50, 65, 80 };
78. FIS_TYPE fis_gMFI1Coeff3[] = { 70, 90, 110 };
79. FIS_TYPE fis_gMFI1Coeff4[] = { 100, 115, 130 };
80. FIS_TYPE fis_gMFI1Coeff5[] = { 120, 140, 165 };
81. FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coeff1, fis
    _gMFI1Coeff2, fis_gMFI1Coeff3, fis_gMFI1Coeff4, fis
    _gMFI1Coeff5 };
82. FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff, fis_
    gMFI1Coeff };
83.
84. FIS_TYPE fis_gMFO0Coeff1[] = { -40, 0, 40 };
85. FIS_TYPE fis_gMFO0Coeff2[] = { 10, 50, 90 };
86. FIS_TYPE fis_gMFO0Coeff3[] = { 60, 100, 140 };
87. FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1, fis
    _gMFO0Coeff2, fis_gMFO0Coeff3 };
88. FIS_TYPE fis_gMFO1Coeff1[] = { -40, 0, 40 };
89. FIS_TYPE fis_gMFO1Coeff2[] = { 10, 50, 90 };
90. FIS_TYPE fis_gMFO1Coeff3[] = { 60, 100, 140 };
91. FIS_TYPE* fis_gMFO1Coeff[] = { fis_gMFO1Coeff1, fis
    _gMFO1Coeff2, fis_gMFO1Coeff3 };
92. FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff, fis_
    gMFO1Coeff };
93.
94. int fis_gMFI0[] = { 0, 1, 0 };
95. int fis_gMFI1[] = { 1, 1, 1, 1, 1 };
96. int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1};
97.
98. int fis_gMFO0[] = { 1, 1, 1 };
```

```
int fis_gMFO1[] = { 1, 1, 1 };
        int* fis_gMFO[] = { fis_gMFO0, fis_gMFO1};

        FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

        int fis_gRType[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

        int fis_gRI0[] = { 1, 1 };
        int fis_gRI1[] = { 1, 2 };
        int fis_gRI2[] = { 2, 1 };
        int fis_gRI3[] = { 2, 2 };
        int fis_gRI4[] = { 2, 3 };
        int fis_gRI5[] = { 2, 4 };
        int fis_gRI6[] = { 2, 5 };
        int fis_gRI7[] = { 3, 5 };
        int fis_gRI8[] = { 3, 4 };
        int fis_gRI9[] = { 3, 2 };
        int fis_gRI10[] = { 3, 1 };
        int fis_gRI11[] = { 1, 4 };
        int fis_gRI12[] = { 1, 5 };
        int* fis_gRI[] = { fis_gRI0, fis_gRI1, fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5, fis_gRI6, fis_gRI7, fis_gRI8, fis_gRI9, fis_gRI10, fis_gRI11, fis_gRI12 };

        int fis_gRO0[] = { 1, 3 };
        int fis_gRO1[] = { 1, 2 };
        int fis_gRO2[] = { 1, 3 };
        int fis_gRO3[] = { 1, 2 };
        int fis_gRO4[] = { 3, 3 };
        int fis_gRO5[] = { 2, 1 };
        int fis_gRO6[] = { 3, 1 };
        int fis_gRO7[] = { 3, 1 };
        int fis_gRO8[] = { 2, 1 };
        int fis_gRO9[] = { 1, 3 };
        int fis_gRO10[] = { 1, 3 };
        int fis_gRO11[] = { 3, 1 };
        int fis_gRO12[] = { 3, 1 };
        int* fis_gRO[] = { fis_gRO0, fis_gRO1, fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5, fis_gRO6, fis_gR
```

```
    O7, fis_gRO8, fis_gRO9, fis_gRO10, fis_gRO11, fis_g
    RO12 };
135.
136.        FIS_TYPE fis_gIMin[] = { 85, 40 };
137.
138.        FIS_TYPE fis_gIMax[] = { 95, 140 };
139.
140.        FIS_TYPE fis_gOMin[] = { 0, 0 };
141.
142.        FIS_TYPE fis_gOMax[] = { 100, 100 };
143.
144.
145.        FIS_TYPE fis_MF_out(FIS_TYPE** fuzzyRuleSet,
     FIS_TYPE x, int o)
146.        {
147.            FIS_TYPE mfOut;
148.            int r;
149.
150.            for (r = 0; r < fis_gcR; ++r)
151.            {
152.                int index = fis_gRO[r][o];
153.                if (index > 0)
154.                {
155.                    index = index - 1;
156.                    mfOut = (fis_gMF[fis_gMFO[o][ind
    ex]])(x, fis_gMFOCoeff[o][index]);
157.                }
158.                else if (index < 0)
159.                {
160.                    index = -index - 1;
161.                    mfOut = 1 - (fis_gMF[fis_gMFO[o]
    [index]])(x, fis_gMFOCoeff[o][index]);
162.                }
163.                else
164.                {
165.                    mfOut = 0;
166.                }
167.
168.                fuzzyRuleSet[0][r] = fis_min(mfOut,
    fuzzyRuleSet[1][r]);
169.            }
170.            return fis_array_operation(fuzzyRuleSet[
    0], fis_gcR, fis_max);
```

```
171.        }
172.
173.        FIS_TYPE fis_defuzz_centroid(FIS_TYPE** fuzz
    yRuleSet, int o)
174.        {
175.            FIS_TYPE step = (fis_gOMax[o] - fis_gOMi
    n[o]) / (FIS_RESOLUSION - 1);
176.            FIS_TYPE area = 0;
177.            FIS_TYPE momentum = 0;
178.            FIS_TYPE dist, slice;
179.            int i;
180.
181.            for (i = 0; i < FIS_RESOLUSION; ++i){
182.                dist = fis_gOMin[o] + (step * i);
183.                slice = step * fis_MF_out(fuzzyRuleS
    et, dist, o);
184.                area += slice;
185.                momentum += slice*dist;
186.            }
187.
188.            return ((area == 0) ? ((fis_gOMax[o] + f
    is_gOMin[o]) / 2) : (momentum / area));
189.        }
190.
191.
192.        void fis_evaluate()
193.        {
194.            FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
195.            FIS_TYPE fuzzyInput1[] = { 0, 0, 0, 0, 0
     };
196.            FIS_TYPE* fuzzyInput[fis_gcI] = { fuzzyI
    nput0, fuzzyInput1, };
197.            FIS_TYPE fuzzyOutput0[] = { 0, 0, 0 };
198.            FIS_TYPE fuzzyOutput1[] = { 0, 0, 0 };
199.            FIS_TYPE* fuzzyOutput[fis_gcO] = { fuzzy
    Output0, fuzzyOutput1, };
200.            FIS_TYPE fuzzyRules[fis_gcR] = { 0 };
201.            FIS_TYPE fuzzyFires[fis_gcR] = { 0 };
202.            FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules,
     fuzzyFires };
203.            FIS_TYPE sW = 0;
204.
205.            // Transforming input to fuzzy Input
```

```
206.            int i, j, r, o;
207.            for (i = 0; i < fis_gcI; ++i)
208.            {
209.                for (j = 0; j < fis_gIMFCount[i]; ++
    j)
210.                {
211.                    fuzzyInput[i][j] =
212.                        (fis_gMF[fis_gMFI[i][j]])(g_
    fisInput[i], fis_gMFICoeff[i][j]);
213.                }
214.            }
215.
216.            int index = 0;
217.            for (r = 0; r < fis_gcR; ++r)
218.            {
219.                if (fis_gRType[r] == 1)
220.                {
221.                    fuzzyFires[r] = FIS_MAX;
222.                    for (i = 0; i < fis_gcI; ++i)
223.                    {
224.                        index = fis_gRI[r][i];
225.                        if (index > 0)
226.                            fuzzyFires[r] = fis_min(
    fuzzyFires[r], fuzzyInput[i][index - 1]);
227.                        else if (index < 0)
228.                            fuzzyFires[r] = fis_min(
    fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
229.                        else
230.                            fuzzyFires[r] = fis_min(
    fuzzyFires[r], 1);
231.                    }
232.                }
233.                else
234.                {
235.                    fuzzyFires[r] = FIS_MIN;
236.                    for (i = 0; i < fis_gcI; ++i)
237.                    {
238.                        index = fis_gRI[r][i];
239.                        if (index > 0)
240.                            fuzzyFires[r] = fis_max(
    fuzzyFires[r], fuzzyInput[i][index - 1]);
241.                        else if (index < 0)
```

```
242.                             fuzzyFires[r] = fis_max(
    fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
243.                         else
244.                             fuzzyFires[r] = fis_max(
    fuzzyFires[r], 0);
245.                     }
246.                 }
247.
248.                 fuzzyFires[r] = fis_gRWeight[r] * fu
    zzyFires[r];
249.                 sW += fuzzyFires[r];
250.             }
251.
252.             if (sW == 0)
253.             {
254.                 for (o = 0; o < fis_gcO; ++o)
255.                 {
256.                     g_fisOutput[o] = ((fis_gOMax[o]
    + fis_gOMin[o]) / 2);
257.                 }
258.             }
259.             else
260.             {
261.                 for (o = 0; o < fis_gcO; ++o)
262.                 {
263.                     g_fisOutput[o] = fis_defuzz_cent
    roid(fuzzyRuleSet, o);
264.                 }
265.             }
266.         }
```

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1. Kesimpulan

Pada tugas akhir ini telah dibuat dan dirancang sistem navigasi pada balon udara menggunakan GPS dan kontrol logika *fuzzy*. GPS digunakan untuk mendeteksi posisi didukung dengan kompas untuk mendeteksi arah hadap. Kontrol logika *fuzzy* digunakan pada sistem pengaturan motor kemudi untuk pergerakan balon. Berdasarkan pengujian beban yang telah dilakukakan didapatkan titik tengah gravitasi balon pada pada jarak 40 cm dari depan balon, 24 cm dari kanan balon, dan 36 cm dari kiri balon. Berat maksimum yang dapat diangkat oleh balon adalah 233 gram. Akuisisi data sensor IMU untuk mengetahui derajat arah hadap menggunakan algoritma AHRS dapat menghasilkan kesalahan pengukuran kurang dari 7%. Pengukuran posisi objek menggunakan sensor GNSS menghasilkan posisi berada pada rentang 9,5-20 meter dari posisi sebenarnya. Penggunaan logika *fuzzy* dapat diterapkan pada sistem kontrol motor kemudi untuk pergerakan balon dengan *variable* yang digunakan berupa galat derajat arah hadap balon udara terhadap derajat *bearing*. Terjadi osilasi untuk mempertahankan arah Utara dengan simpangan berkisar antara 50-90 derajat dan arah Barat dengan simpangan sebesar 20-50 derajat.

5.2. Saran

Beberapa saran yang penulis bisa berikan untuk pengembangan tugas akhir adalah penambahan sistem navigasi berbasis inersia agar pergerakan posisi balon udara dapat lebih tepat dan penambahan algoritma *differential* GPS agar pengukuran posisi sensor GNSS menghasilkan koordinat yang akurat.

DAFTAR PUSTAKA

[1] J. Rao, Z. Gong, J. Luo and S. Xie, "A flight control and navigation system of a small size unmanned airship," in *IEEE Int. Conf. Mechatronics and Automation*, Canada, 2005.

[2] P. Chaklos, "RC Blimp, Remote Control Airship, Inflatables," Above & Beyond Inc, 15 Januari 2010. [Online]. URL: http://www.advertisingballoons.com/remote-control-blimps.htm. [Diakses 15 Juni 2016].

[3] Ozzmaker, "BerryIMU-accelerometer, gyroscope, magnetometer," Tindie, Inc., 11 April 2014. [Online]. URL: https://www.tindie.com/products/ozzmaker/berryimu-accelerometer-gyroscope-magnetometer/. [Diakses 15 Juni 2016].

[4] K. H. Rob O'Reilly, "Sonic Nirvana: MEMS Accelerometers as Acoustic Pickups in Musical Instruments," Analog Devices Inc, 1 Juni 2009. [Online]. URL: http://www.sensorsmag.com/sensors/acceleration-vibration/sonic-nirvana-mems-accelerometers-acoustic-pickups-musical-i-5852. [Diakses 15 Juni 2016].

[5] W. Storr, "Hall Effect Sensor and How Magnets Make It Works," Electronics-tutorials.ws, 14 April 2016. [Online]. URL: http://www.electronics-tutorials.ws/electromagnetism/hall-effect.html. [Diakses 15 Juni 2016].

[6] W. R. Hamilton, Elements of Quaternions, Longmans: Green & Company, 1866.

[7] P. Hapala, "Quaternion," Wikimedia Foundation, Inc., 11 November 2007. [Online]. URL: https://en.wikipedia.org/wiki/Quaternion. [Diakses 16 Juni 2016].

[8] F. Dai, W. Gao, N. Kushida and L. Shang, "Fuzzy control for the autonomous airship," in *6th IEEE Conference on Industrial Electronics and Applications*, Beijing, 2011.

[9] L. Brits, "Conversion between quaternions and Euler angles," Wikimedia Foundation, Inc, 11 November 2007. [Online]. URL: https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles. [Diakses 15 Juni 2016].

[10] C. Chamberlain, "Understanding Euler Angles," CH Robotics LLC, 06 Juni 2009. [Online]. URL: http://www.chrobotics.com/library/understanding-euler-angles. [Diakses 15 Juni 2016].

[11] S. O. H. Madgwick, A. J. L. Harrison and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, Zurich, 2011.

[12] J. Velagic, N. Osmic, F. Hodzic and H. Siljak, "Outdoor navigation of a mobile robot using GPS and GPRS communication system," in *ELMAR, 2011 Proceedings*, Zadar, 2011.

[13] M. Looney, "Inertial Sensors Facilitate Autonomous Operation in Mobile Robots," *Analog Dialogue Web Archives,* vol. 44, 2010.

[14] N. H. M. P. Peter J. Ersts, "The Perpendicular Distance Calculator," American Museum of Natural History, Center for Biodiversity and Conservation, [Online]. URL: http://biodiversityinformatics.amnh.org/open_source/pdc/documentation.php. [Diakses 15 Juni 2016].

[15] T. Sutojo, E. Mulyanto and V. Suhartono, Kecerdasan Buatan, Yogyakarta: Andi Offset, 2011.

[16] S. Kusumadewi, Artificial Intelligence (Teknik dan Aplikasinya), Yogyakarta: Graha Ilmu, 2003.

[17] R. AS, Modul Pembelajaran Rekayasa Perangkat Lunak, Bandung, 2011.

BIODATA PENULIS

Dimas Arief Rahman Kurniawan lahir di Kebumen pada 6 Januari 1994, yang merupakan anak kedua dari empat bersaudara dari pasangan Dulchaeri dan Sri Sukarti. Penulis menyelesaikan pendidikan dasar di SDN Berkoh 3 Purwokerto dan dilanjutkan dengan pendidikan menengah di SMPN 8 Purwokerto dan SMAN 1 Purwokerto. Pada tahun 2012, penulis memulai pendidikan di jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah, penulis aktif dalam kegiatan robotika dan aktif sebagai asisten laboratorium Elektronika Dasar.

Email : dimasark@yahoo.co.id